A Cognitive Diagnostic System for Explaining Algebra Errors

Heather MAYS

B. Sc. (Hons.), Grad. Dip. Ed., M. Sc., Grad. Dip. Computing

Thesis submitted as total fulfilment of the requirements for the degree Doctor of Philosophy

School of Information Technology & Mathematical Sciences,

University of Ballarat,

P. O. Box 663,

Gear Avenue,

Mt. Helen,

Ballarat, Victoria, 3353,

Australia.

September, 2001

Table of Contents

List of Figures

List of Tables

List of Figures

Abstract

This thesis presents a new approach to cognitive diagnosis within the domain of algebra that has greater explanatory power than existing techniques. Previous approaches have achieved high levels of success when modelling the error-making processes for procedural problems, ie. those that can only be solved by a single solution technique (such as multi-digit subtraction). This success resulted from the fact that the solution of such problems can be decomposed into a single set of skills, the execution of which is then analysed. However, for problems that can be solved in a variety of ways, it is important that the student's chosen technique is identified because different techniques require different skills and hence give rise to different sources of errors. If the solution technique is not identified, then cognitive diagnosis is limited to simply determining that an individual student has not mastered the given type of problem but cannot provide an explanation of how the student interprets it and attempts to solve it. This, in turn, limits the capability of an interactive learning environment in tailoring remediation and instruction to the needs of the individual.

Underpinning this new approach is the taxonomy of errors that was developed as a result of analysing the error-making process in terms of the related solution technique. Although the literature in the area of mathematics education contains details of common algebraic errors, these had not previously been classified in a manner that easily enabled a computerised system to link an erroneous answer with the solution technique that led to it. In this research, erroneous answers that were given to a particular problem were clustered according to the technique that was applied to the problem. Analysis of the answers was then conducted to determine the structural commonalities of the answers within each group and the structural differences between the groups. The results of the analysis revealed that the structure of an answer is a strong indicator of the solution technique that was used to generate the answer. This outcome provided evidence that partial matching of answers would allow for efficient retrieval of similar cases and hence case-based reasoning would be a suitable implementation paradigm for the diagnostic system.

Evaluation of the research was conducted by designing and testing a cognitive diagnostic system. The diagnostic system was implemented using a two-step process. Firstly, questions are classified at the time of entry. Each case in the *Question Type* library is represented by an exemplar problem. The *Solution Technique* case base is the collection of all these exemplars and it is this case base that is used during the second operational phase (ie. the diagnostic phase). Each case in the library contains all common erroneous answers for the given problem type, a description of the solution technique that led to it and a generative mechanism to adapt the given problem type in terms of the query problem. The generative mechanisms thereby enable the system to distinguish between slight variations on a particular solution technique. This methodology means that the system can dynamically classify problems and generate the link between a question (as entered by the teacher) and the answer given by a student.

The results of the evaluation indicated that the system accurately classified problems. This was expected because of the well-defined nature of the domain, that is, whilst there are infinitely many variations that a given problem type can take, there are only limited numbers of these problem types. The results of the evaluation also indicated that the diagnostic system could accurately diagnose errors for problems that can be solved by a multiplicity of approaches, but was no better than existing systems when diagnosing errors on problems that can only be solved by a single technique.

Chapter 1 Introduction

Whilst the last twenty years has seen a dramatic increase in the production of computerised teaching/learning systems, there has been a concomitant *decrease* in the research interest accorded to cognitive diagnosis. The result of this situation has been the development of systems that can determine whether or not the answer given by a student to a particular problem is correct, but that cannot adequately **explain** the error-making process. This, in turn, makes it very difficult for on-line systems to tailor remediation to the needs of an individual. The current research attempts to redress this situation in the domain of algebraic problem solving.

To achieve this aim, three main research questions had to be addressed:

1. How to approach cognitive diagnosis so that the resultant system would have greater explanatory power than existing systems,
2. How to categorise errors in a manner that could be used as the basis for the diagnostic process, and
3. How to represent knowledge of students' problem-solving and error-making processes in the most appropriate manner.

The outcomes of this research have been tested by designing and implementing a cognitive diagnostic system for tertiary entry-level algebra that can automatically determine a learner's misconceptions from their one-line answer to an algebra question. The errors made on a particular problem are then **explained** in terms of the solution technique that the student applied to the problem.

This chapter provides an overview of the different requirements of the cognitive diagnostic system and outlines how the thesis is organised.

1.1 Motivation

Over the last ten to twenty years, universities around the world have identified a common problem - the declining level of mathematical understanding of students at tertiary entry level. This problem impacts on the ability of students to construct new mathematical knowledge and on their ability to make efficient use of tools such as computer algebra systems, which have become common-place in educational institutions at both the secondary and tertiary levels. Simultaneously, the sizes of the intake cohorts at universities have increased and the cohorts have shown a greater diversity in terms of the ages, backgrounds and ability levels of the students.

A common approach to redressing the problem has been the introduction of diagnostic testing of students entering tertiary studies, particularly in Science and Engineering courses. In many cases, the term “diagnostic” testing is something of a misnomer, because the results of the tests have been used simply to determine the most appropriate units in which students should enrol; this type of testing should really be called “intake” testing. To warrant the name “diagnostic” testing, the results of the test need to be collated and analysed with the aim of tailoring some form of follow-up (or remediation). Performing this task by hand is a very slow process. For this reason, there has been a shift in delivery mode from hand-written tests to on-line tests. Typically, on-line tests adopt one of two main approaches - using either hard-coded multiple choice questions or a bank of hard-coded questions for which common erroneous answers have already been determined. Very few systems exist that are dedicated to diagnosing errors (probably the most successful of these is *DIAGNOSYS*, which is discussed in detail in the next chapter). Those systems that do exist suffer from limited **explanatory** power; that is they can identify but not attribute causes to erroneous answers. As a result, the output from such systems is limited to identifying broad curriculum areas that have not been mastered.

There is a need for a more dynamic assessment scheme that caters for the idiosyncratic behaviour of individual users. This can be achieved by constructing a knowledge-based system that is fully automated and that has some capability of

intelligent reasoning about the error-making process in algebra. Such a system requires the ability to retrieve items of information and perform similarity matching on them. It is therefore proposed that the use of case-based reasoning will provide a suitable paradigm for implementing a diagnostic system.

1.2 Cognitive Diagnosis

"Cognitive diagnosis is the attribution of errors in the performance of cognitive tasks to causes in terms of *deficient* or *confused* knowledge" (De Koning *et al*, 1995). As such, cognitive diagnosis is an important component of the student modelling task within an interactive learning environment (ILE). The student model is the component of an ILE that contains knowledge of the processes of teaching and learning, which is used to adapt tuition and remediation to the needs of the individual user (Greer and McCalla, 1994).

Initial attempts at diagnosing errors on procedural tasks (such as multi-digit subtraction) produced excellent results by using rule-based systems and bug catalogues to identify student errors. Similar results have been achieved within the domain of algebra using a skills-based approach, but only for problems that have a single available solution technique, eg. multiplying a bracketed term by a signed number. For problems that have a multiplicity of available solution techniques, cognitive diagnosis becomes a much more complex task for two main reasons: the skills required to solve a problem are dependent upon the chosen solution technique and identifying the solution technique can lead to a combinatorial explosion of possible answers. Due to these complexities, cognitive diagnosis is an area of research that has received decreased attention in recent years; the attention that it has received has largely been for the purpose of updating the student model. Other reasons given for this fall from grace include the fact that "human teachers do not (overtly) engage in cognitive diagnosis … that ILEs can operate successfully without it and that is very difficult (if not impossible) to realise full cognitive diagnosis" (Self, 1992).

However, "..a more important role of cognitive diagnosis than providing data for updating the student model is in providing the topics to remedy, and to **explain** to the student what has gone wrong or what has led to an impasse. The adaptive competence of an ITS is directly proportional to its discriminative and valid diagnostic power" (De Koning *et al*, 1995). Therefore, the aim should be to construct systems that **complement**, rather than mimic, the performance of human tutors (particularly inexperienced ones). The difficulties experienced with cognitive diagnosis should not preclude some attempt being made nor should the system be expected to **completely** solve this problem. Instead, a cognitive diagnosis system should provide human tutors with information about problems experienced by individual students, problems experienced by entire classes, the nature of errors that are commonly made by students and the solution techniques that students most commonly adopt. This information can help the human tutor to direct their classroom practices. From a student perspective, the feedback can help them to **reflect** on their approaches to problem solving with the aim of developing improved knowledge structures, better links between these knowledge structures and rules for checking their success in problem solving.

The characteristics of useful cognitive diagnosis (Self, 1996) include:

- it should incorporate techniques for hierarchically decomposing domain knowledge from general abstraction at the highest level to detailed forms that can be expanded as required,
- it should be based on a *series* of observations, but should evolve during a session to improve search efficiency,
- it should distinguish between slips (which can be considered to be a failure in execution rather than in intent) and genuine misconceptions,
- it should provide a set of candidate explanations that are ranked in order of likelihood, and
- it should involve the learner interactively rather than passively.

The current research has focused on the design and development of techniques that can improve the granularity of cognitive diagnosis provided by a computerised

system, using the above recommendations as a guide. Fundamental to this is the manner in which errors are categorised, and hence the current research has adopted a new approach to error analysis.

1.3 Error Analysis

Most systems that attempt to diagnose errors in algebraic problem solving employ a catalogue of bugs. This has been made possible by the outcomes of research into the learning of algebra that has been conducted in the classroom (which is described in detail in the next chapter). However, this method of approaching cognitive diagnosis has not led to the production of systems with a satisfactory level of explanatory power. For example, the types of erroneous solution techniques that students generate and that were identified by Matz have not been easily accommodated by existing systems (Matz, 1980, 1982).

The current research has adopted the approach of identifying the relationship between the **structure** of a student's one-line answer to an algebra problem and the solution technique that produced the answer. A distinction is made between a generalised technique (ie. a solution **plan**) and the number of minor variations that can arise from it when a single step is executed in different ways (ie. skill errors). For problems that have a multiplicity of available solution techniques, the set of erroneous answers generated by a single solution plan were grouped to identify commonalities within the set of answers and differences between the sets of answers (see chapter four for details). This work led to a taxonomy of errors and a related taxonomy of algebra problems. However it also raised a new question: How could an automated system determine what a problem is asking so that it could make the link between a wrong answer for a particular problem and erroneous answers given by past students on a non-identical problem of the **same** type? The answer was provided by developing a model of algebraic problem solving that engages in the same processes that a student does.

1.4 Modelling Algebraic Problem Solving

Systems that use hard-coded questions have a direct link between the questions and all the common erroneous answers. However, an automated system does not and therefore such a system must be capable of deriving this link for itself. This is the same problem that the student faces, and hence by building the system around a model of human algebraic problem solving, it enables the system to identify (some of) the conceptual activities undertaken by the student, not just the procedural tasks.

There is broad agreement between cognitive scientists about the manner in which students approach mathematical problem solving. The method is similar to the four-stage model of general problem solving devised by Polya (Polya, 1948). The four stages are:

1. Interpret - match a familiar problem stored in memory (the base) with the new (or target) problem.
2. Plan - identify the steps taken to solve the base problem and adapt these to the target problem.
3. Execute - carry out the activities associated with the base problem.
4. Review - determine whether the adapted solution plan was successful in solving the target problem and why.

This model has been adapted to the task of diagnosing algebra errors - it is the “glue” that holds the different system components together (see chapter 3 for full detail).

1.5 The Cognitive Diagnostic System

The outcome of the current research is a cognitive diagnostic system for algebra that has greater explanatory power than existing systems. The system is capable of:

1. Decomposing an algebra problem in a manner that enables it to classify the problem in terms of the steps required to solve it. This means that the system is not limited to using a set of problems that have been previously analysed nor does it need to store **multiple** algebra questions of the same type.
2. Decomposing an erroneous one-line answer to an algebra problem in a manner that enables it to identify the solution technique that was most probably adopted by the student. The diagnoser identifies a **set** of possible diagnoses, which are ranked by the similarity scores between the problem under investigation and others that have been encountered in the past.
3. Identify and explain errors in terms of the misconceptions held by the student (as exemplified by the use of either inappropriate or suboptimal solution techniques). Students can be very creative in producing their own solution procedures, not all of which are legitimate. Identifying the adopted solution technique provides information to the human tutor as to the manner in which students interpret problems and plan their answers.
4. Identify and explain errors made by the student when executing a solution technique. Student problem-solving performance is an unstable process - students can both learn and forget, and their performance can be affected by other factors such as context (Van Lehn, 1989 and Payne and Squibb, 1990). It is therefore important that the human tutor can identify the points at which this performance breaks down.

1.6 Organisation of the Thesis

The remainder of this thesis is divided into seven chapters. The content of each of these chapters is now outlined.

Chapter two provides details of the background to this research, which includes two major areas: the learning of mathematics (in particular the domain of algebra) and the application of artificial intelligence to the construction of systems in the area of mathematics education. The first area that is discussed is the findings from a series of classroom studies into the approaches that students take when learning and using algebra. These studies are the Long Term Study conducted in the US during the period 1974-1978, the Concepts in Secondary Mathematics and Science project conducted in the UK during the period 1974-1979, and the follow-up study called Strategies and Errors in Secondary Mathematics conducted in the period 1980-1983. Related to these works are the topics of human memory and knowledge representation. Particular emphasis is placed on the outcomes of the Long Term Study and the proposal of schemata and critics to explain both knowledge representation and problem-solving expertise. Linked to these topics are theories of human problem solving in the domain of mathematics and the modelling of error-making within the domain.

The second major research area relevant to the current work is the application of artificial intelligence to the construction of mathematical teaching/learning systems. Two cognitive architectures that have been proposed to model different theories of cognition and that have been used to underpin teaching systems in a variety of domains (viz. the ACT architecture and the SOAR architecture) are discussed as well as several systems dedicated to the diagnosis of errors in mathematics (these are *DEBUGGY*, the Leeds Modelling System and *DIAGNOSYS*). Two implementation paradigms (rule-based reasoning and case-based reasoning) are then compared in terms of their ability to not just identify but also to explain the error-making process. This is followed by a justification for the choice of paradigm used here (viz. case-based reasoning). Finally, an alternative approach to cognitive diagnosis is proposed.

Chapter three introduces the model of algebraic problem solving used as the basis for the design of the current cognitive diagnostic system. This model was derived from theories of the way in which students represent their mathematical knowledge and the impact of this knowledge representation on students' perceptions of similarity in the

domain of algebra. Related to these theories is the nature of problem solving expertise and how this is manifested during the four main stages of problem solving, viz. the interpretation or classification of problems, solution planning, the execution of the chosen solution plan and finally reviewing the results of the problem-solving process. Finally, the model of algebraic problem solving and details of how it is implemented in the current system are presented.

Chapter four presents the details of the analysis conducted on the algebraic error data collected from the Diagnostic Test conducted at the University of Ballarat in the period 1996-1999. The purpose of this work was to adopt an approach to error analysis that could help to realise the aim of finer-grained cognitive diagnosis that has been achieved by existing systems within the current domain. Particular emphasis is placed upon the relationship between a student's chosen solution technique and the form of their final answer. It is also shown that students, particularly early learners, can show great inconsistency in their problem-solving performance; this is manifested in their choices of solution techniques. Several factors that were found to affect the choice of solution technique (viz. the degree of recognition required to choose the most appropriate technique, the degree of detail of a question, the student's self-confidence and their level of problem-solving expertise) are then detailed.

Chapter five provides details of the design of the diagnostic system, which is based upon the model of problem solving presented in chapter three. This begins with the definition of the domain and some architectural limitations that were imposed on the system. An overview of the system is then provided in terms of its requirements and the two main phases of its operation (ie. setting a test and answering a test). Because the system has been implemented using case-based reasoning, similarity matching is fundamental to its success; this point is addressed in terms of how the system represents algebra questions and measures the similarity of cases and how it represents answers and measures the similarity of two answers. The impact of this work is then discussed in terms of the design of the two case-based reasoners (ie. the *Question Type* reasoner and the *Solution Technique* reasoner) and the manner in which case

retrieval is achieved. Finally, discussion of the maintenance and extension of the system is included.

Chapter six outlines how the design of the diagnostic system has been implemented and identifies and describes the processes and functions required to achieve this. An overview of the system architecture and data flows through the system are provided. The details of case representation provided in chapter five identified the need for a system component that decomposes questions and answers. This component is the parser and its operation is detailed. A second component, the marking module, also required its own methods which are described here. Because the system has two main phases of operation (the classification of questions and the diagnosis of erroneous answers), the details of system implementation (such as the construction of queries, the similarity measures and retrieval algorithms used) are discussed in terms of the separate tasks. Unlike the classification task, the second task, diagnosis, involves all the stages common to case-based systems, particularly the adaptation and reuse of exemplar cases. The processes required to achieve this are also described in detail. Finally, the performance of the system on a small run-time test is demonstrated.

Chapter seven discusses the performance of the diagnostic system in its dual tasks of question classification and error diagnosis. The first section discusses formative evaluation and the prototypes that were developed as the system design evolved as well as an experiment that was conducted using neural networks to perform the classification task. The remainder of the chapter is dedicated to the evaluation conducted on the final system, including an assessment of how well the current system meets the needs of a cognitive diagnostic system, measures of the system's performance (correctness, precision, recall, efficiency and consistency) on each of the two main tasks and discusses potential improvements and future directions of the research.

Chapter eight outlines the contribution made by this research, the achievements and limitations of the diagnostic system and provides a summary of the thesis.

Chapter 2 Background and Related Work

This chapter presents the background to the current research. It begins with a description of the main limitations of existing cognitive diagnostic systems (viz. they provide only coarse-grained diagnosis and they have limited explanatory power) and a discussion of the need for improved cognitive diagnosis in the domain of mathematics education, which is the focus of this thesis. The motivation for the current research is that universities from around the world have identified a number of problems experienced by students when making the transition from secondary- to tertiary-level mathematics. These problems are therefore discussed, along with a review of the approaches that universities have adopted in an attempt to redress the situation. The development of the new approach to cognitive diagnosis is based upon research findings from two broad areas, viz. mathematics education and artificial intelligence. Hence, the remainder of this chapter discusses relevant outcomes from research into mathematical problem solving, the error-making process, human memory and knowledge representation, automated reasoning and its application to cognitive diagnosis.

2.1 Justification for a New Approach to Cognitive Diagnosis

Research into computerised educational systems has been an active area for the last twenty years, during which time the name has changed from computer-based instruction through intelligent tutoring systems to interactive learning environments. The variety of names reflects the changes in educational philosophy and practice; for simplicity we will

use the term *intelligent learning environment* (ILE) to denote such a system. In general, the structure of an ILE includes: content (notes, examples, problems and test) and a module to control the interaction (the student model). Initially, the development of computerised tutoring systems focused on the materials for inclusion, and hence the systems were only capable of delivering teaching materials in a manner and order prescribed by the developer of the system. Since the advent of such systems, research has focused on the application of artificial intelligence to the modelling of student performance, with the aim of making the systems behave like an expert classroom teacher who is capable of adapting their behaviour to the requirements of the individual user (Greer and McCalla, 1994).

The aim of ILEs is to provide a one-to-one supplement to classroom teaching. The Student Model is the component of an ILE that contains knowledge of the processes of teaching and learning, which is used to adapt tuition and remediation to the needs of the individual user. Initial attempts at student modelling used a history of the student's performance to achieve this. In other words, these models recorded the questions that a student attempted, the student's responses and whether or not each response was correct, but they did not attempt to diagnose any errors. The second generation of student models incorporated libraries of bugs (or mal-rules) to detect the errors that a student made when solving problems. These models focused on *procedural* errors (those made when executing a step in the solution plan) and did not attempt to diagnose the misconceptions underpinning the errors. Studies of algebraic problem solving have shown that mal-rules on their own do not provide sufficient information about the student, because an individual's problem-solving behaviour can be inconsistent and so the application of mal-rules is unstable (Payne and Squibb, 1990). For these reasons, the latest generation of student models attempts to extend the capabilities of earlier models so that they are capable of capturing (at least some of) the *cognitive* processes that students undertake when solving problems, not just the *manipulative* ones. Examples of these models have been implemented in a variety of ways, including dynamic updating of relational databases (Kuzmycz, 1993), representing problem states as a set of constraints (Ohlsson,

1994) and using Bayesian analysis to evaluate mastery of rules and skills (see Van Lehn *et al*, 1998 and Mitrovic, 1998).

The evolution of student modelling is leading to the development of systems that are better able to emulate several important characteristics of an expert classroom teacher (see Sison and Shimura, 1998, for a discussion of this point). The first of these is well-structured knowledge both of the domain and of the processes of teaching and learning. The latter knowledge type has proven to be the more difficult to capture and implement. However, it is this knowledge that produces other desired characteristics of teaching including adaptability (the ability to represent knowledge in a variety of ways to improve a student's mental models) and responsiveness (tailoring teaching techniques to the needs of an individual). Finally, apart from diagnosing and correcting misconceptions, a teacher can help a student to make the transition from novice to expert problem solver.

Cognitive behaviour is a knowledge-based process, but behaviour on a specific task is not a function of the student's complete knowledge base (Wachsmuth, 1988). Rather, the knowledge applied to a given task will be activated by contextual information both from the learning situation and from the problem task. It is well recognised that an individual's problem solving behaviour is not static but can regress under cognitive load. In other words, the student can move *backwards* along the expertise continuum and this issue needs to be addressed by the student model. The issue of inconsistency in knowledge application is crucial because the discovery of inconsistencies can lead to the identification of flaws in the student's knowledge base and points for remediation. The aim of instruction is to bring about knowledge that is widely applicable, but achieving this requires knowledge of preferred solution strategies and shifts in the level of expertise. To improve its knowledge of a student's expertise, an ILE must monitor as many as possible of the processes that students undertake when solving problems, particularly in multi-step problems that involve changes of goal at different stages. These types of problems are most likely to induce degradation in performance because of the increase in cognitive load produced by the extra control processes required.

Initially, cognitive diagnosis was considered to be the most important element of student modelling; in fact, the term cognitive diagnosis was used interchangeably with the term student modelling in early research into interactive learning environments. However, cognitive diagnosis was also recognised as being very difficult to effect (for example, Self 1991). Later attempts at student modelling therefore tended to shift the emphasis from cognitive diagnosis to other attributes including affective factors such as motivation and self-concept, and conative factors, which are concerned with the student's wants, intentions and learning style (see Self, 1994). This shift in emphasis was driven by several observations, viz. ILEs can perform satisfactorily without involving cognitive diagnosis, human teachers do not (overtly) spend much time in diagnosis, it is very difficult, if not impossible, to maintain a complete and accurate model of the student and, finally, cognitive diagnosis is only used to direct remediation (Self, 1996). However, Self addressed these observations and stressed the need to revisit the area of cognitive diagnosis and to develop rigorous methods for achieving it. In particular, he noted that cognitive diagnosis need not be restricted to remedial purposes only. The fact that classroom teachers do not overtly spend much time in diagnosis does not mean that they do not undertake such activities. Even if it were the case that human teachers never undertake diagnosis, computerised systems should not be restricted to emulating human performance, instead they should complement it. Whilst cognitive diagnosis is difficult to achieve, Self maintained that this should not be a reason to abandon it - he agreed that it is impossible to maintain a complete and accurate model of the learner, since cognitive diagnosis models a moving target (the student can learn or forget knowledge and students' problem-solving behaviours are notoriously unstable, particularly early learners who have limited knowledge structures). However it is still important to identify student difficulties, particularly those that are recurrent, because they represent genuine misconceptions on the student's behalf.

One way in which the effectiveness of teaching and learning can be gauged is through assessing students' problem-solving skills, whether this is in a classroom situation or

within a computerised system. The aim is to help students to improve their mental models and their discrimination between different problem types and available solution techniques. Factors that affect problem-solving performance include the manner in which the subject has represented their knowledge internally and their level of problem-solving expertise (see chapter 3 for more detail). These characteristics are reflected in the answers that students give to questions (see chapter 4 for more detail). Cognitive diagnosis is concerned with ascribing reasons for the errors that people make during problem solving, and is one element of the task of student modelling within a computerised tutoring system. Identifying a student's solution technique is fundamental to the accurate diagnosis of any misconceptions that they hold, because the solution technique is the path between the given question and the student's answer (Ohlsson and Langley, 1988 and Ohlsson, 1994). Misconceptions manifest themselves as errors, but appropriate remedial action can only be formulated once we are reasonably certain of how the errors arose.

When solving any novel problem, we consult memory to determine whether we have encountered a similar situation in the past. If so, we retrieve the previous problem and its solution, and then attempt to adapt this solution to fit the new situation. If there are major differences between the past and present cases, these can be used to guide the representation of the new case in memory and to form links between the different cases, because learning is a continual process of adapting and reorganising our internal knowledge structures. Cognitive scientists agree that when students are presented with a mathematical problem, they will use a form of analogical reasoning to plan a solution strategy by matching the current problem situation with a familiar one stored in memory (English, 1997). They must plan a solution strategy because even a very simple algebra problem can be approached in a variety of ways. However, this "plan" may either be incomplete or simply be to perform a reflex action such as expanding bracketed terms. To access their stored knowledge, students must form a mapping between the current problem and existing representations in long-term memory. Once the mapping has been created, the stored problem can then be recalled along with its solution, which can be adapted to fit the new problem. This mapping is achieved in two stages that are based

upon the goal of the problem, the problem environment and different problem states achieved during the solution process. The first stage of the planning process is a parsing or categorisation problem. The inputs to this stage are the goal (or keyword) plus a set of surface features of the problem, whilst the output is some representation of the question that can then be compared with the student's own stored knowledge representations to identify a suitable solution strategy. Which strategy the student chooses will depend upon their beliefs (ie. their interpretation of the problem and what is required to solve it) and their familiarity with operations which can be applied to achieve the necessary steps (Genesereth, 1982). These points are discussed in more detail in the next chapter.

Different solution strategies expose the student to different possible sources of error (Birenbaum *et al*, 1993). Consider the following problem: "*Solve for* x: $7-2x=1$". Two valid solution paths are outlined in Table 2.1.

Table 2-1 Two valid solution paths for solving a linear equation

Method A	Method B
$7-2x=1$ $7=1+2x$ $\Rightarrow 7-1=2x$ $\Rightarrow 6=2x$ $\Rightarrow \frac{6}{2}=x$ $\Rightarrow x=3$	$7-2x=1$ $\frac{7}{2}-x=\frac{1}{2}$ $\Rightarrow -x=\frac{1}{2}-\frac{7}{2}$ $\Rightarrow -x=\frac{-6}{2}$ $\Rightarrow -x=-3$ $\Rightarrow x=3$

The skills required to solve the problem, and hence any errors made, are determined by the solution process employed. For example, by rewriting the equation with the x term on the right-hand side in method A, the student completely avoids the use of both directed number and numerical fractions, which are fundamental to method B. Therefore, it is imperative that if a computerised system is to be capable of identifying a student's errors and of diagnosing their underlying misconceptions, it must first be capable of identifying the student's solution strategy. The error analysis conducted as part of the current research

showed that it is possible to infer a student's most probable solution strategy from a single-line answer (Mays et al, 1997); this work is presented in detail in chapter 4.

As a second example, consider the question *Factorise* $(x+3y)^2 - y^2$. The keyword (*Factorise*) should trigger something in a student's mind about potential solution techniques. Commonly chosen techniques for a problem of this type are:

- application of the difference of two squares **template** $x^2 - y^2 = (x-y)(x+y)$,
- application of the **quadratic formula** $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$,
- application of other factorisation **heuristics**,
- **expansion** of the bracketed term, collecting like terms and then factorising the resultant expression, and, erroneously,
- application of "**generalised distributivity**" $(x+3y-y)^2 = (x+2y)^2$ (Matz, 1980).

The reason why these techniques are chosen lies in how students categorise problems and their familiarity and facility with the available solution techniques (which may include invalid options such as generalised distributivity).

So why do existing systems underachieve in the diagnosis of algebraic errors? The answer is two-fold. Firstly, attempting to ascribe reasons for a student's errors requires that the system be based upon a model of algebraic problem solving and error-making, which is not the case for existing rule-based systems. Secondly, although algebraic errors are well documented, they have not previously been categorised in a manner that enables a system to **explain** the reasoning that led to a particular erroneous answer. The current research attempted to remedy this situation by developing a model of algebraic problem-solving, a taxonomy of errors to underpin the operation of a diagnostic system and methods for determining the most likely solution path that the student followed. Further, the implementation of the system required a paradigm that has greater explanatory power than rule-based systems and that can utilise sophisticated partial-matching techniques

when attempting to match a student's answer with others resident in the library. For these reasons, case-based reasoning was chosen as the implementation paradigm.

In summary, this research is based upon relevant findings from the areas of mathematics education and artificial intelligence. Therefore, the next section outlines the problems that universities have identified with students entering tertiary-level mathematics, and the remainder of this chapter reviews research into the learning of algebra and the solution processes used by students, human mathematical problem solving, error-making during problem solving, human memory and knowledge representation and the application of artificial intelligence to mathematics education.

2.2 Diagnostic Testing and Remediation in Tertiary Mathematics

Over the last two decades, educationalists from around the world have noted that the standard of mathematical understanding of entry-level students has been steadily declining. This phenomenon appears to be independent of teaching styles, learning styles and other factors such as geographical location, as evidenced by the source of these papers. In Australia, the list of researchers includes Barrington and Carbone, 1991, Swedosh, 1996, 1999, Frid, 1997, Swedosh and Clark, 1997, 1998, Caccetta *et al*, 1997a, 1997b, and Barry and Davis, 1999. In the United Kingdom, research has also been presented by a number of authors including Sutherland and Pozzi, 1995, Kitchen, 1996, Rycraft, 1997, Anderson *et al*, 1998 (who claimed that the situation only gets worse by the time that students reach the third year of their studies), Larcombe, 1998, Gill, 1999, and Kitchen, 1999. There have been similar reports from other countries such as Germany (Kurz, 1985 and Kurz and Hohloch, 1996) and South Africa (Bezuidenhot, 1998).

The area of mathematics that has received the greatest attention is algebra. There are several reasons for this. Firstly, algebra is the language of mathematics and its applications, and hence is fundamental to studies in science and engineering. Secondly,

algebra is not usually part of the standard tertiary curriculum (although it is often included as part of remedial mathematics units), but it is important that students be able to identify and correct their own misconceptions and incorrect solution techniques. Thirdly, the advent of computer algebra systems has seen radical changes being made to tertiary mathematics curricula (Pozzi, 1993, Yearwood and Glover, 1993, Yearwood *et al*, 1995 and Larcombe, 1996). As computer algebra systems become part of the everyday toolkit for students in tertiary mathematics classes, it is crucial that students understand the *structures* and *concepts* underlying mathematics if they are to make efficient use of the tools.

Various authors have shown that algebraic deficiencies can be overcome by the use of diagnostic tests and targeted remediation (for example, Edwards, 1997, Swedosh and Clark, 1997). Therefore, in an attempt to address problems with entry-level students, many universities have introduced diagnostic tests for students entering science and engineering courses. In this section, we review some of the approaches that universities have taken to address problems that students experience with algebra, and discuss the findings in terms of what assistance a computerised system can offer.

2.2.1 Diagnostic Tests

Although many universities are changing their mode of diagnostic test delivery from pen-and-paper to on-line delivery, this is typically done to make use of the speed of correction and the immediacy of feedback and to accommodate large group sizes (in some universities, the test may be delivered to cohorts comprising over a thousand students). On-line systems also have the advantage that students can take tests as often as they choose and whenever they choose; that is, by delivering the test on-line universities can be more flexible in meeting the needs of their students. What does *not* change from one mode of delivery to the other is the type of questions included on the test or the level of diagnosis provided to the students. One notable exception to these comments is the *DIAGNOSYS* system developed and used in the United Kingdom. For this reason, we pay particular attention to this system. Before that, we review the practices adopted by one

Australian university, viz, the University of Melbourne in Victoria. This university is a well-established institution and its intake is very large and can be expected to comprise students who have achieved high entry scores. Despite this last point, lecturers at the university felt the need to introduce diagnostic testing for entry-level students and have reported their results extensively in the literature.

2.2.1.1 Diagnostic Testing in Australia - the University of Melbourne

Intake testing has been used for determining unit enrolments at the University of Melbourne for many years. However, in 1989, the School of Mathematics introduced a diagnostic test for all students entering Science and Engineering courses, because there was a perceived need for extra support for low-ability students (Barrington and Carbone, 1991). The cohort of students at the University of Melbourne is traditionally one of the best, as measured by entrance scores gained at Year 12. The particular group under consideration was in the top thirteenth percentile of the 1992 intake to all Victorian universities (Swedosh, 1996) and comprised about 400 students.

The diagnostic test was divided into four sections - algebra, graphing, calculus and trigonometry - and all questions were short answer. Initially the answers were simply marked as being correct or incorrect but, after compiling the results, lecturers conducted analysis of student workings on the diagnostic test. The results of this analysis were later compared with the end-of-year examination taken by the students, and with the results from La Trobe University (another of the major Victorian universities). The aim of the analysis was to identify common misconceptions and their frequency of occurrence. Despite the fact that the intakes to both universities had been adjudged to be amongst the best in the state, students exhibited many of the errors that have been observed in the literature (see section 2-5). Table 2-2 contains details of the most common misconceptions.

Table 2-2 Common Misconceptions on the University of Melbourne Diagnostic Test (Swedosh, 1996)

Question	Observations
Factorise $(2x+y)^2 - x^2$	27% gave the answer $3x^2 + 4xy + y^2$. The students expanded rather than factorised even though they had successfully answered earlier questions involving factorisation.
Simplify $\log_{10} 45 + \log_{10} 2 - \log_{10} 15$	13% of students gave the answer $\log_{10} 32$, that is they used the malrule $\log A + \log B = \log(A + B)$. This error is a member of the family known as *Generalised Distributivity*
Solve for x $\{x: 2x + 4 < 5x + 10\}$	13% gave the response that $-6 < 3x$ (correct) but simplified this to $x < -2$. They did not reverse the inequality when dividing by a negative number. A further 6% gave the response $x = -2$.
Solve for x: $\frac{3}{x} - \frac{4}{a} = \frac{5}{b}$ where a and b are positive, real numbers.	25% appear to have first inverted each of the 3 fractions without finding a common denominator. This seems to indicate that they believe that anything can be done to an equation as long as the same thing is done to both sides. Here that has meant taking the reciprocal of each term rather than each side of the equation. This is an example of what Matz called *Repeated Application* errors.

Solve for x $x^3 - 3x + 2 = 0$ given that $x = 1$ is one solution	Equation was rewritten as $x^3 - 3x = -2 \Rightarrow x(x^2 - 3) = -2 \Rightarrow x = -2$ *or* $x^2 = 1$, with a frequency of 20 - 25%. This is an extrapolation of the null factor rule, where the student does not understand the significance of the zero on the right-hand side of the rule.
Simplify $\dfrac{x^2 - 5x - 6}{x^2 - 2x - 3}$	There were many different incorrect answers given, but two of the most common involved incorrect cancellations to yield: $\dfrac{-5x - 2}{-2x - 1}$ and -3*x*. Students tended to believe that any single term in the numerator could be cancelled with any single term in the denominator.
Simplify $2^{-1} + 3^{-1}$	About 24% of the students put either 5^{-1} or 5^{-2}.
Simplify $2^x + 2^x$	The most common incorrect answers here were 4^x and 2^{2x}.
	Also in solving a trig equation which required use of the identity cos^2 A = 1 - sin^2 A, there appeared to be the common misconception that $x^2 = C^2$ has only one root, namely $x = C$.

Explanations for these misconceptions have been discussed in the literature (see Matz, 1980 and 1982, Tirosh 1990 and Vinner 1990) and in section 2.5 of this thesis. Insufficient understanding of mathematical concepts can cause an inappropriate transfer of theorems or adaptation of existing concepts; that is students over-generalise incorrect or inadequate rules. To overcome this, teachers need to discuss the students' misconceptions with them. The follow-up work undertaken with this cohort of students is discussed in section 2.2.2.

Similar work has been conducted at other Australian universities for similar reasons. One of these is the Curtin University of Technology, where the Department of Mathematics made two decisions in the mid-1990s: to introduce diagnostic testing for entry-level students, and to use information technology wherever appropriate to improve teaching and assessment practices (Caccetta *et al*, 1997a). The number of students sitting the test each year is in the order of 500. With such a large number of students. the decision was taken to develop an on-line test that should include (at least) algebra, trigonometry and calculus. By using an on-line test, immediate feedback can be given to the student and for a number of reasons Curtin decided to develop their own testing shell (MQUEST), which is available on the internet. It contains a bank of hard-coded questions in multiple-choice format (see Caccetta *et al*, 1997b). Students are presented with 25 questions chosen randomly from the test-bank. Once these have been marked, the system advises the student about areas of mathematics that the student needs to address, as well as the units in which they are advised to enrol. The system does not provide diagnosis beyond this level.

Several approaches to the use of diagnostic testing in the United Kingdom are now discussed.

2.2.1.2 Diagnostic Testing in the United Kingdom - DIAGNOSYS

The Teaching and Learning Technology Programme (TLTP) was set up in the United Kingdom in 1992 to enable a coordinated approach to be taken to addressing a number of issues regarding the emergence of the use of information technology in teaching and learning in tertiary institutions. The TLTP funded a vast array of projects, particularly in the area of interactive learning environments aimed at helping students to make the transition from school to university. Since then, the United Kingdom has been the several projects, each with the aim of developing diagnostic testing systems for mathematics, including the CALM project at Heriot-Watt University (see Beevers *et al*, 1995),

Mathletics at Brunel University (see Greenhow, 1996), MCQ at Nottingham and Keele Universities (see Brydges and Hibberd, 1994, Hibberd, 1996 and Quinney, 1997) and *DIAGNOSYS* at Newcastle University (see Appleby, 1994, Appleby and Anderson, 1997 and Appleby *et al*, 1997). These tests range from pen-and-paper tests, through computer-generated multiple choice tests to a knowledge-based diagnostic system (*DIAGNOSYS*) that is capable of handling open-ended questions and includes an algebraic input facility. This system is "... the only expert-system test designed specifically for mathematical subjects ..." (Appleby, 2000) and is now discussed in detail.

In the early 1990s, a team of lecturers from Newcastle and Leeds Universities produced the first prototype of *DIAGNOSYS* - a knowledge-based computer diagnostic test. It was developed for university entry-level students in Engineering programmes, but can be adapted for other groups. The project was motivated by declining entry standards and increasing drop-out rates in the Engineering programme at Newcastle University. Because of large class numbers, it was decided that a computer-based test would be more efficient than a hand-marked test, as it would provide instantaneous feedback to students and make collation of group results faster. Individual reports are provided for each student to aid them in identifying their strengths and weaknesses, whilst analysis of the group profile is used by the lecturing staff each year to assist in course design.

Rather than using a multiple-choice test, the developers wanted to be able to produce a test that had open-ended questions and that could be adapted to the student's level of entry knowledge (as measured by factors such as the course to be studied, the highest level of mathematics previously studied and the grade achieved) and their performance on the test. To enable students to easily input their answers, a mathematics interface was created. The development of the system consisted of several strands: a skills network (i.e. the hierarchy of skills in a given knowledge domain), question design (based on the skills network), a test shell (that can be applied to other domains eg. a test in mechanics has been created using *DIAGNOSYS*), an expert system which generates the student profile, the mathematics interface and tutorial for its use, utility programs which produce

feedback to lecturers and follow-up materials. The skills network was determined by lecturers and comprises 92 skills at four levels of difficulty considered to be part of the algebra domain (*cf.* Birenbaum *et al* 1993). The network is divided into a number of component topics, and is set up as a hierarchy with directed links between related skills. This network is used to maintain a student profile and to determine which questions should be asked (part of the skills network is reproduced in Figure 2.1). Each skill is tested with one or two questions of the same style and level of difficulty. The authors made the following observation about the use of hierarchies of skills to model the acquisition of mathematical expertise.

"*The use of hierarchies for organising the process of skill acquisition was first proposed by Gagne and comes from the behaviourist tradition. Whilst the authors of DIAGNOSYS do not necessarily hold to this educational view (as it has significant implications for the way the educational context of the Test and the support material is perceived), they believe that the use of a hierarchy is sufficiently valid to be useful in the assessment of certain kinds of cognitive skill, especially the procedural skills prevalent in basic mathematics. The links present within the DIAGNOSYS network are not intended to describe all these prerequisite associations. The number of links has been optimised so as to balance the number of questions asked against the reliability of the inferences made, leading to some of the prerequisite associations being omitted.*" (Appleby, 2000)

The network is used in the following manner. If skill A is considered to be subordinate to skill B, then if a student shows mastery of skill B the system infers that the student has also mastered skill A (consider the level 2 skill 210 *Expand* ...(...) and the level 3 skill 312 *Expand* (...)(...) in Figure 2.1). Before running the test, the student enters their highest maths qualification and this is used to initialise the network, i.e. each question is marked as either "possibly known" or "unasked". A student can have up to 5 "lives" on each question. During the running of a test, questions are chosen by using a set of inferencing rules to traverse the network of skills. If a question for a given skill is answered correctly, then the system infers that all linked prerequisite skills are "probably known", and dependent skills, which are usually at a higher level, are set to "possibly

known". If a question is answered incorrectly, then it is inferred that all dependent skills are also probably unknown. The chosen questions are ones that the system has inferred that the student possibly knows, but which have no dependent skills marked as "possibly known".

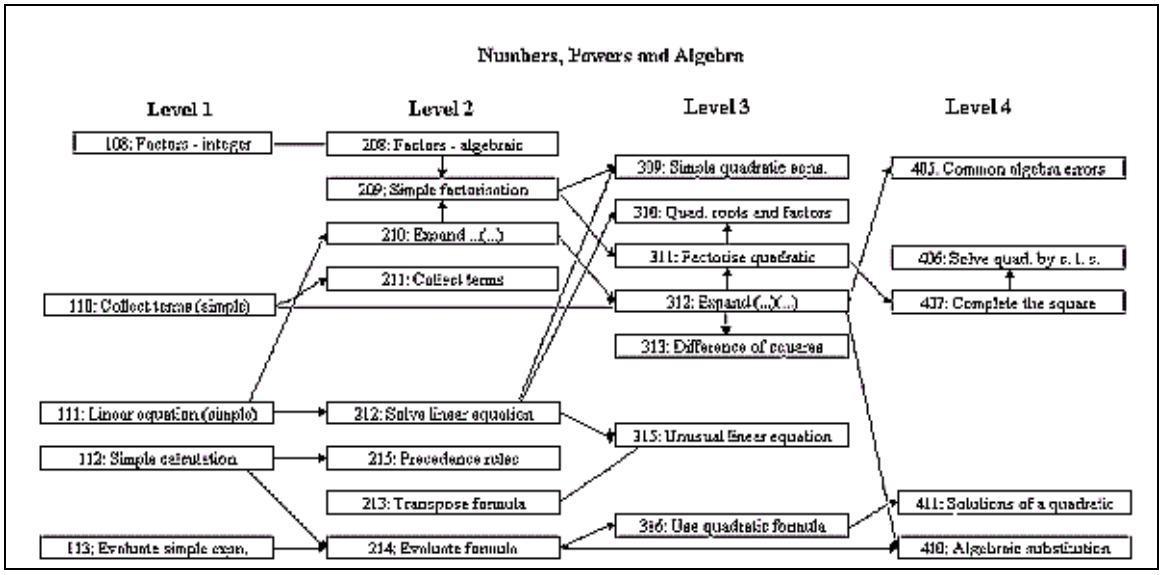

Figure 2-1 Part of the Skills Network used in DIAGNOSYS (Appleby, 2000)

The test is terminated either when there are no questions left to be asked or when a time limit is reached. There are three possible sources of error in this inferencing procedure: when a student correctly answers a question by guessing (this is much more likely to be a problem in systems that use multiple-choice questions), when a student knows the correct answer but enters it incorrectly (particularly for open-ended questions) and when the links between skills are not valid. This last point has been addressed during the production of *DIAGNOSYS* by use of statistical analyses of student responses. The developers found that the inference rules in an earlier version of the system had a success rate of 84% (Appleby, 2000) and used these results to update both the questions and the skills network.

However, whilst it may be true that one skill is subordinate to another, it is not necessarily the case that the ability to answer a particular question is a prerequisite for being able to answer another. The inference rules also fail to take account of the inconsistencies that students exhibit during problem solving (these points are addressed in the next section). When a student completes a test, the results are written to a text file that is used to generate reports: one for the individual student and one for the teacher. The student report includes the student's name, the date, the total number of questions the student answered, the number of questions the student answered correctly, a breakdown of the skills included on the test and the level of mastery achieved by the student on each skill, and general comments. The results are presented in both text and graphical formats. An example of a student report is included in Figure 2.2.

There are five options for the format of the reports provided to the teacher: a group profile of all the students tested (see Figure 2.3), a ranked listing of the students by total score, tabulated answers actually given to all questions by all students (to highlight common misunderstandings), results of all questions and skills for subsequent spreadsheet analysis and a combined file of individual profiles ready for printing. These programs are used for two main purposes: designing courses to fit the needs of the cohort, and identifying those students who may be in need of extra help. By including common, erroneous answers in the analysis, the developers can identify particular areas that require intervention by the teacher.

Name: Appleby

Date: 5 9 2000

Questions asked: 37

Questions correct: 23

Total mark (on ALL questions) 74%

****

TOPICS

numbers 100%

powers 80%

algbasic 100%

algmethods 50%

equations 50%

miscell 67%

graphs 40%

areavol 56%

****

GENERAL COMMENTS

You should note any weaknesses in your basic mathematics. Your tutor will advise you on additional work needed, and which books or materials to use.

Time used: 18

Lives: 5

Lives used: 5

Lives used successfully: 3

Figure 2-2 A student profile in *DIAGNOSYS* (Appleby, 2000)

Columns are %:

1: tested on topic (total)

2: successful on topic (total)

3: qual{A-level}

4: qual{AS-level}

5: qual{Other}

		1	2	3	4	5
101	Multiplication of negative numbers	75	75	100	0	100
102	Multiply negative and positive	75	75	100	0	100
103	Negative Numbers	75	75	100	0	100
203	Definition of negative powers	100	100	100	100	100
204	Rules for positive powers	100	100	100	100	100
205	Inverse ratios	75	75	100	0	100
206	Cancelling numerical fractions	75	75	100	0	100
210	Expanding one bracket	75	75	100	0	100
211	Collecting terms	75	50	100	0	0
212	Solving linear equations	75	75	100	0	100
312	Expanding two brackets	100	75	100	100	0
313	Difference of squares	100	50	50	100	0
314	Simultaneous Equations	100	0	0	0	0
411	Solutions of a quadratic	75	50	50	100	0
441	Product rule	25	25	0	100	0
442	Integration of powers	25	25	0	100	0
Total in each column:		4	4	2	1	1
Average mark:		68	62	84	65	
Average time used:		7	10	3	5	
Average lives used:		3.0	2.5	2.0	5.0	
Av used successfully:		1.8	1.5	1.0	3.0	
Av questions asked:		32	25	34	44	
Av questions correct:		23	18	28	30	

Figure 2-3 A group profile in *DIAGNOSYS* (Appleby, 2000)

DIAGNOSYS is probably the most successful system of its type and, because it is a shell, it can also be adapted to other domains that are hierarchical in structure. However the

error diagnosis is coarse-grained, meaning that the system only identifies broad areas for remediation. It does not attempt to identify the solution techniques adopted by a student, and does not take account of inconsistencies in problem-solving performance. For example, the network contains a Level 3 skill "Factorise quadratic" and a Level 4 skill "Solve quadratic by completing the square", but no other distinction is made between the approaches that can be adopted to solving quadratics. As shown in chapter 3, there are multiple approaches that can be adopted and these vary in their level of difficulty. In fact, the technique "Difference of Two Squares" is easily recognised for problems presented in the form a^2 - b^2, but not for problems that contain more detail (such as $16(4a-3b)^2-25b^2$). This means that the methodology adopted in *DIAGNOSYS* is not suitable for the purpose of fine-grained diagnosis and student modelling in an interactive learning environment.

At this point, we consider some of the follow-up support that universities provide based upon the results of their diagnostic tests, to determine which of these remediation techniques could be incorporated in our own diagnostic system.

2.2.2 Remediation and Follow-up Support

The aims of diagnostic testing vary across institutions. At the lower end of the scale, the test data are used purely to determine the most appropriate units in which each student should enrol. This type of testing is more accurately termed *intake* testing. For testing to be accurately called *diagnostic* testing, there must be some sort of follow-up and remediation in place. To achieve this, the data collected from a diagnostic test need to be analysed to identify long-term trends in an individual's growth of mathematical understanding, to improve the educator's knowledge of their students and, subsequently, to aid in long-term planning for both secondary and tertiary curricula (Kurz, 1985 and Frid, 1997). Several authors have stressed the need to provide remedial courses that are designed to match the needs of the courses in which the students are enrolled (for example, Edwards, 1997). Further, these courses need to emphasise mathematical *methods*, not just content because, if students are to improve their level of mathematical expertise, they need to be encouraged to consider different approaches to problem

solving, and to develop critical knowledge about the applications and limitations of the different approaches. This requires the student to review all aspects of the problem-solving process, viz. facts and skills, problem representation, planning and monitoring, and the development and application of verification procedures. "*In summary, the research indicates that strategic skills are more likely to be acquired when instruction includes a fine-grained approach to the use of particular strategies that are discussed and applied within specific content areas*" (Geiger and Galbraith, 1999).

At Bournemouth University, follow-up support is provided in the form of an Extra Maths unit (Edwards, 1997). This unit is optional for students scoring 40% or better on the diagnostic test, but is compulsory for those students who scored less than 40%. The diagnostic test is administered to all students enrolled in the first year of Engineering courses. There are two main strands in the course - Product Design, which contains no higher mathematics, and Technology, which contains calculus (although there is no prerequisite for an entering student to have studied preparatory units in secondary school). Because the courses vary dramatically in their content, so do the content and duration of the Extra Maths units taken by students enrolled in the different strands.

The study revealed that for students enrolled in the Design course, their marks on the Diagnostic Test were significantly correlated with their marks on the end-of-year exams for their analytical subjects. At first glance, this may indicate that the Extra Maths unit had no impact on the subsequent performance for these students. However, attendance at the Extra Maths unit was observed to decline as the year progressed for a variety of reasons including other, and more immediate, assessment requirements, the fact that students did not earn credit for the Extra Maths unit and the availability of supplementary examinations for core units. For those students who *did* attend regularly, significant improvement in their performance was realised. Similar results were observed for the cohort enrolled in the Technology course. The conclusion drawn by the author was that while attendance at Extra Maths is no guarantee of success in improving mathematical

performance of students, appropriate remediation *can* be successful provided regular reviews are conducted to ensure that the support is effective.

One of the most disturbing features of algebraic errors is their persistence. Students at entry-level to tertiary studies still display some of the misconceptions that are demonstrated by early secondary-level students (Swedosh, 1996 or Swedosh and Clark, 1997). To address this situation, the remediation provided at the University of Melbourne is based upon the "conflict teaching approach", which involves teachers discussing misconceptions with the students in an attempt to help students to identify their own inadequacies (Tirosh, 1990 and Vinner, 1990). In the follow-up study, students enrolled in the same first-year mathematics were divided into two homogeneous groups: an experimental group of 90 students who received conflict teaching and a control group of 50 students who received standard teaching. The two groups covered the same material, and were taught at the same pace. Based upon the results of the diagnostic test, the experimental group were confronted with the most common misconceptions (such as $\frac{1}{x} = \frac{1}{a} + \frac{1}{b} \rightarrow x = a + b$) and correct but incomplete answers (eg. $x^2 = 81 \rightarrow x = 9$). Most of the verification techniques used were numerical, eg. $2 + 3 = 5$ *but* $\frac{1}{2} + \frac{1}{3} \neq \frac{1}{5}$, but some focused on applying alternative solution techniques, for example consider the following: $x^2 = 81 \Rightarrow x^2 - 81 = 0 \Rightarrow (x + 9)(x - 9) = 0 \Rightarrow x = \pm 9$.

A post-test was then given to both groups. The results of this test revealed that the treatment group exhibited both a significant increase in the proportion of correct answers and a significant decrease in the proportion of responses that included the identified misconceptions. Follow-up interviews with some students from the treatment group indicated that while their first reaction to solving a problem was still error-prone, they had improved their error-checking procedures and self-corrected many of the original mistakes, i.e., the immediate effect of the conflict teaching approach was successful in reducing the incidence of the common misconceptions (Swedosh and Clark, 1998).

These results then raised the question of how persistent the improvement would be. Therefore, a follow-up study was conducted with a subgroup of the original treatment group (Swedosh, 1999). Participation was on a voluntary basis (some of the students had advanced to second-year mathematics units while others were no longer studying any mathematics) and the same set of questions was used. The results of the research showed that, while some students had reverted to their earlier methods, overall the group showed *significant* and *sustained* improvement from their first diagnostic test. Swedosh was careful not to extrapolate the results beyond the group under investigation. However, he concluded that the conflict approach was successful in reducing the incidence of misconceptions with students who were "strong mathematically". This observation can be expected because students who are strong mathematically are more likely to monitor their progress during problem solving and to reflect upon their answers than are students who are less able (see chapter 3). The success of the conflict approach in the classroom suggested that it would provide a useful basis for remediation in a diagnostic system.

This section has detailed problems encountered by students entering tertiary studies in mathematics and the methods that universities have adopted to remedy the situation including one cognitive diagnostic system (i.e. *DIAGNOSYS*). Currently, there is a dearth of systems that can both deliver fine-grained diagnosis and explain the reasoning that led to the diagnosis. Two other approaches to the problem are compared in the next section.

2.3 Applications of Artificial Intelligence to Mathematics Education

The two most common approaches to developing student models in interactive learning environments are the mal-rule approach and the model-tracing approach (Nwana, 1993). A mal-rule system usually employs a rule-based system in the diagnosis of student misconceptions based upon a single answer to a question. Known errors are encoded in a library (or database) and are linked to the hard-coded questions. In this way, there is a direct link between the student's erroneous answer and associated errors. On the other

hand, a model-tracing system guides the student through the system's model of problem-solving. This means that the system is capable of solving a particular problem itself and may have multiple approaches available. However, if the student attempts to use an approach that has not been recognised by the system, it will interrupt the student.

Both approaches have short-comings. ILEs built on the mal-rule approach do not engage the student in diagnosing and remediating their own inadequacies whilst those built on the model-tracing approach are too prescriptive and do not allow for student creativity in solving complex problems. This means that if a student has adopted a valid solution technique which has not been hard-coded into the system, the technique will be adjudged by the tutoring system to be incorrect. This discourages students both from using the system and from being creative in their approach to problem-solving. Instead, the tutoring system should be able to check the validity of the student's argument and add it to its own repertoire. This requires a different approach to the modelling of student performance.

One such method is the problem-space method (Ohlsson and Langley, 1988, Ohlsson, 1994) that is based upon a model of problem solving (Newell and Simon, 1972). The basis of the method is to identify a path that leads to the given answer and then to construct a procedure that can generate the path. The vastness of false knowledge can result in combinatorial explosion, so, to limit the search for a solution path, the system uses psychological principles to generate constraints. This method has been used to implement tutors in arithmetic (for subtraction and addition, see Ohlsson, 1994) and SQL (Mitrovic, 1998). The latter tutor does not contain domain knowledge *per se*, due to the problems with natural language processing. Instead the rules of the domain are used to develop a set of problem-solving constraints and the resulting student model comprises the set of constraints that were violated by the student during problem solving. The model is subsequently used to direct the delivery of content to the individual, but does not achieve cognitive diagnosis. The performance of the arithmetic tutors is discussed in section 2.3.3.

In this section, we reviewed the results achieved by other systems that have applied one of the above techniques to the task of student modelling and error diagnosis in the domain of mathematics. The first of these is the mal-rule approach and the two systems discussed are DEBUGGY (in the domain of multi-digit subtraction) and The Leeds Modelling System (in the domain of algebra), the second approach that is discussed is model-tracing and several examples based on the ACT architecture are outlined, while the final method that is examined is Constraint Based Modelling and the system that is investigated is the Automated Cognitive Modeler (ACM).

2.3.1 Mal-rule Systems

One method for achieving cognitive diagnosis in a computerised system is to have a very limited set of questions and associated answers (correct and incorrect) hard-coded into the system. When a student inputs their answer it is compared with those available within the system. If there is an exact match between the student's answer and one resident in the library, then we can be reasonably certain that we can reuse the information associated with the answer (eg. remediation). However, when there is no direct match, it requires the knowledge engineer to intervene to incorporate the new answer into the system and, possibly, to change the entire structure of the system (Sleeman, 1984). In this section, we review the success of two diagnostic systems; one in the domain of multi-digit subtraction and the other in the domain of algebra.

2.3.1.1 DEBUGGY

DEBUGGY is a rule-based diagnostic system for multi-digit subtraction developed by John Seeley-Brown and Richard Burton during the late seventies (Brown and Burton, 1978 or Brown and Van Lehn, 1980). It was based on an earlier system called BUGGY, which was used with student teachers to help them in identifying bugs and systematic errors in their students' work on subtraction. Many errors that students make are systematic in that they appear to be the result of incomplete or incorrect learning. These are referred to as "bugs" and can be predicted from a student's performance on related

questions. Other errors (or “slips”) are unsystematic and seem to be the result of carelessness.

The basic premise of the researchers was that students are highly competent at following a procedure, but that they often make errors because they follow the *wrong* procedure. Therefore, the most important aspect of cognitive diagnosis is the identification of the student’s procedure. DEBUGGY was based upon a procedural network of the knowledge underlying the skills required to solve the problems. The model can be represented as a graph with procedures on the nodes and relationships on the arcs. Each node is divided into a conceptual part (which represents the purpose of the procedure) and an operational part (which includes details of the methods required to perform the purpose). This knowledge was hard-coded in the system, and implemented using a rule-based approach.

In DEBUGGY, diagnostic tests were developed and trialled on 925 students who were learning subtraction. The purpose was to see if the system could correctly diagnose bugs and slips for these students. DEBUGGY proved to be at least as accurate as human diagnosticians in determining bugs that individual students displayed. Some students were retested two days after the initial test to measure the short-term stability of bugs and others were retested several months later to test for the long-term stability of bugs. It was shown that bugs are not stable and that about one third of student errors could not be categorised as either bugs or slips. In summary, this work showed that students can vary in their choice of a problem solving strategy depending upon the problem context.

Repair Theory was developed to account for this variability (Brown and Van Lehn, 1980). The theory is based upon the observation that when a student reaches an impasse in problem solving, (s)he will attempt to proceed by adapting a strategy that has proved successful in other situations. These attempts are referred to as “repairs”. In general, repairs do not lead to a correct solution. However, Brown and Van Lehn did find that repairs also follow a pattern and so are themselves predictable. This indicates that it is important to identify the problem solving strategy adopted by a student to fully

understand why errors are made, particularly in a domain such as algebra where nearly all questions can be approached in a number of different ways that require different sets of skills. Another observation was that the behaviour of the model (in terms of its accuracy and simplicity) could be affected by the choice of a language for knowledge representation (van Lehn, 1982).

2.3.1.2 Leeds Modelling System

The second system reviewed here is the Leeds Modelling System (LMS) created as a hybrid diagnostic/teaching system for algebra skills, which incorporated domain knowledge, a student model, a set of teaching tasks and a rule set that related the student model and the teaching tasks (Sleeman, 1982, 1984, 1985). The structure of the system was similar to that of BUGGY in that it incorporated a generative mechanism to create diagnostic models. The basis of the system was a collection of recognised "mal-rules" or incorrect procedures observed during student problem solving (these can be found throughout the literature; for an example, see Payne and Squibb, 1990).

The first version of LMS attempted to generate a diagnostic model (i.e. to identify student misconceptions), but did not attempt remediation. An underlying assumption was that once a student had correctly applied a rule, they would always use the rule on subsequent tasks. However, this assumption inhibited the diagnostic ability of LMS. Removing the assumption led to an explosion in the number of rules to be considered and so a reformulation, both of the rule database and the system architecture, was conducted.

The later version of LMS was tested using a group of 14-year-olds and follow-up pen-and-paper tests were also conducted to help in verifying the diagnoses produced by the system. LMS was capable of identifying most of the common bugs but also found that students were unstable in their choice of solution techniques. In particular, Sleeman noted that whilst more able students were capable of creating and experimenting with their own solution procedures, less able students only attempted to use methods with which they were familiar, but that they had difficulty in choosing between them. This helps to explain

the inconsistencies in their problem-solving performance (using different techniques for the one problem in different contexts). The analysis also led Sleeman to observe that students acquire a common set of bugs and impasses, indicating that there is a specific mechanism common to all students. Hence, there are clearly identifiable families of errors (similar to those proposed by Matz), but LMS was incapable of working with these.

In summary, the mal-rule approach has been successful in modelling the error-making process on **procedural** skills (such as subtraction), but has limited capability when diagnosing errors for problems that can be solved by a variety of solution techniques. This means that this approach is useful when we know exactly what skills are required to solve a problem, but it does not lend itself to diagnosing **conceptual** errors made by a student when choosing a solution technique.

2.3.2 ACT Theory and Model-tracing Systems

Anderson has been one of the most prolific researcher into cognitive theories (a small sample of his work relevant to the current research include Anderson, 1976, Anderson, 1983, Anderson, 1990, Anderson, 1993a, 1993b, Anderson, Reder and Simon, 1995, Ritter and Anderson, 1995, Anderson and Matessa, 1997, Anderson and Lebiere, 1998). His work has resulted in a theory of cognition that has been used to underpin teaching systems in a variety of domains, including mathematics and computer programming. The cognitive architecture that Anderson developed is the ACT theory and later revisions called ACT-R and ACT-R*. In this section, we review the results of applying the architecture to several teaching systems.

The ACT-R theory involves the compilation of knowledge in human memory and its use in problem solving. Knowledge compilation is the process by which the new information is recorded in memory and Anderson believes that this is achieved by the individual creating new production rules. The theory also states that when humans are presented with a new problem, they attempt to recall a similar situation from the past (either from memory or some external source) and to use the past experience to solve the current

problem by analogy. Matching a current problem to an existing one in memory may lead to the retrieval of several competing possibilities. The person thus needs a means of measuring the similarity between the contenders and of choosing the most likely candidate. Typically similarity is a function of the surface features of the problem.

The ACT theory has been applied to the development of a number of production systems that model **ideal** performance on a task in the domain of mathematics. The theory has been adapted in a number of ways to enable the different systems to model different aspects of student performance in mathematics (including error-making, the use of analogy and the identification of student strategies in solving linear equations). These are now discussed.

2.3.2.1 The Equation Solving Tutor

The Equation Solving Tutor is an interactive learning environment in the domain of solving linear equations (for example, Solve for x: $ax + b = c$). The experiment involved students from three first-year algebra classes at Pittsburgh High Schools (Ritter and Anderson, 1995). The tutor is a model-tracing system that contains an expert system capable of solving the problems presented to the student. The model is used to trace the student's actions during problem solving. The purpose of the experiment was to enable researchers to investigate how the removal of the need to perform some low-level skills affected a student's choice of problem-solving strategy. Solving the problem involves two sub-tasks: division and subtraction, hence there are only 2 solution strategies available (-÷¸ and ÷-).

At each step of the solution process, the tutor offers a menu of possible actions to the student, some of which may not be relevant to the problem in hand. The Equation Solving Tutor (EST) can operate in either of 2 modes. In Mode 1, the student plans the solution strategy, but the computer performs calculations (thereby removing the possibility of a student performing an arithmetic error which can affect the difficulty of solving the problem), whereas in Mode 2, the student plans the solution strategy and also performs all

calculations. The major focus of the research was to determine whether or not the requirement to perform arithmetic computations affects students' problem-solving strategies.

According to ACT theory, productions are selected based on the expected utility of a strategy where the utility is itself a function of the probability of the success of the production. This led to the generation of four predictions regarding the success of the experiment.

1. Initial bias may be toward - ÷ as a strategy, but this would be no stronger in either group.
2. Since ÷ - can lead to the use of fractions, students in mode (2) would encounter problems in solving the equation.
3. Students in mode (2) would learn to avoid the ÷ - strategy, but students in mode (1) would return to it because the computer is successful in applying the strategy.
4. Students in mode (1) who were successful with the ÷ - strategy would be less successful on follow-up pen-and-paper tests.

Students interacted with the system by choosing a solution strategy from a menu; if an incorrect action was nominated, the EST beeped the student. Ritter and Anderson observed that students operating in mode 1 (i.e. no-calculate) were more likely to divide first than were students operating in mode 2 (i.e. who were performing their own calculations). These students were then at a disadvantage when they had to perform their own calculations, due to a learned bias towards using a strategy that is less effective in the new context. Students who operated in mode 2 (ie. those who performed their own calculations) were permitted to use a (non-symbolic) calculator to decrease the possibility of making arithmetic errors. The emphasis was thereby placed on the strategy chosen by the student, rather than upon their calculations.

The results of the investigation showed that success in performing an action increased its likelihood of being reused by a student with a new problem, which supported the ACT

theory. It was also found that students who used the computer to perform calculations were indifferent between techniques which had different degrees of difficulty, whereas students who had to perform their own calculations showed a bias for simpler techniques over more difficult ones. All four predictions were confirmed by the experimental results. The observed shift in strategy (from - ÷ to ÷ -) was in accordance with ACT theory. The shift depended on the decrease in probability of reaching the goal following successful application of a production, rather than a failure of the production itself.

2.3.2.2 The Modelling of Errors due to Slippage

Another application of ACT-R was to the modelling of errors due to "slippage" during the performance of a high-level cognitive task, viz. solving simple linear equations in a single unknown while memorising a digit span (Lebière, Anderson and Reder, 1994). Slippage refers to non-systematic errors during problem solving and are usually associated with an increase in the cognitive load of a particular task. To achieve this end, the ACT theory was redefined to create a hybrid system that could accommodate partial matching of information, because systems that are completely rule-based are too deterministic to be able to model degradation of problem-solving performance induced by an increased cognitive load. Connectionist models, such as neural nets, are usually applied to this type of error modelling.

In the experiment, the authors investigated errors of omission and commission in student productions induced by the interference of other memory requirements (memorising the digit span). Errors of omission pertain to the case when a required chunk cannot gather enough activation to be retrieved, whereas errors of commission are concerned with the retrieval of one or more incorrect chunks. The subjects were required to memorise a digit span (of 2, 4 or 6 digits), then to solve a linear equation involving either one or two steps (that is equations of the form $x + a = b$ or $x*m + a = b$, respectively), and finally to recall the digit span. In ACT-based systems, recall of a chunk of information from long-term memory is only effected when it gathers enough activation.

The researchers investigated both types of errors, but found that most of the errors made involved either retrieving the wrong data or performing an incorrect operation; that is, errors of commission dominated errors of omission. To model errors of commission, the authors replaced the usual succeed/fail tests that ACT uses for pattern-matching by a set of constraints to be maximised in terms of the activation measure. This change enabled the researchers to account for errors both quantitatively and qualitatively. The justification for this change in the underlying theory was presented by the authors as follows.

"*There are independent motivations for introducing a partial-matching mechanism into ACT-R beyond accounting for errors of commission. In many situations softer matching is required. Often objects that we are looking for are not exactly like their pattern specification. For instance, people will wear hats or grow beards and we still need to recognize their identity. One might argue that only in formal domains like mathematics does a single mismatch disqualify an item from being useful.*

Error modeling provides an additional advantage. ACT-R is a very general system which allows for many ways to model a particular task. Having to match not only the subjects' rule-like behavior but also their occasional errors provides an additional constraint on the form of the model. In this case, for example, we had to switch from a production-based representation of algebraic transformation rules to a declarative chunk-based representation because the former model could not produce the fan effects necessary to account for the difference in error frequencies among arithmetic operations.

By using ACT-R's concept of activation, not just as a heuristic measure of the likelihood of a match, but as a measure of the match itself, we have shown that the occasional, gradual pattern of errors characteristic of human performance can be effectively modeled by production systems."

These statements beg the question: Why use a production system for modelling error-making when other paradigms such as case-based reasoning or neural nets are, by nature, better able to perform partial matching and to replicate degradation in performance? This point is revisited in section 2.7.

2.3.2.3 Use of analogy

Another adaptation of ACT theory was undertaken to develop a system that was capable of replicating analogical transfer during problem solving (Anderson and Thompson, 1989). The new system was called PUPS (Pen-Ultimate Production System) and was a rule-based system that used an object-oriented approach to knowledge representation, that is, the knowledge representation was much closer to schema theory than production theory. The objects had slots for three types of information: the *category* to which a particular object belongs, the *function* that the structure fulfils and the *form* of the structure. The domain under investigation was the writing of blocks of LISP code to perform a particular function.

The general framework that PUPS used for the production of analogies was a seven-stage process.

1. Obtain a goal problem. An example could be to write a recursive algorithm for the factorial function $n! = n.(n\text{-}1)!$.
2. Find an example similar to the current problem. For the factorial example, a useful "similar" problem could be the summorial function $n_s = n + (n\text{-}1)_s$.
3. Elaborate the goal. The subject will usually have some interpretation of the problem that can be viewed as a set of assertions. During the problem-solving process, the subject may add to, remove or modify these assertions (ie. elaboration shares some characteristics with understanding and others with searching).
4. Generate a mapping between the goal and the example. For a successful analogy to be developed, this step entails distinguishing the nature or purpose of the different slots.
5. Use the mapping to fill in the goal pattern. It is important that all slots are instantiated, whether the slot is a category, function or form slot.

6. Check the validity of the solution.
7. Generalise a form and a summarisation rule. This step mimics the learning that a subject achieves by accommodating new knowledge.

The PUPS system was capable of creating analogies and storing these as productions of the form IF {there exists a target structure needing a particular form and serving a specified function AND there exists a model structure of the desired form and that performs the required function} THEN {try to map the model form onto the target form}. Supplementary domain knowledge can facilitate the second step, viz. the search for a suitable model. If multiple candidates are available, ACT uses activation patterns to identify the best candidate. Activation is functionally dependent upon a number of factors, including the recency and frequency of use, the number of pathways leading to a particular structure and the types of features of different objects. The researchers found that subjects would use models that appeared similar to the target (whether the features were relevant or not) or that had a high usage frequency. They also found that analogies based on these factors were very persistent (that is, students would reuse these models whenever possible). This is in line with student performance in mathematics where erroneous methods, such as generalised distributivity, can be very difficult to dislodge from a student's repertoire.

In the three applications of ACT theory to mathematics education, there were a variety of adaptations required before the theory could achieve the desired performance. Changes such as those in knowledge representation and matching functions indicate (to this author at least) that whilst ACT can be "tinkered" with to mimic different aspects of problem solving, it does not represent the best approach to modelling mathematical problem solving and diagnosing errors. The basic premise, that people use productions to represent knowledge and statistical analysis to determine activation, has not been borne out by observation of human performance. Whilst ACT (or a variation) can reproduce some results of observed human performance, it does not follow the same method that humans do. The changes required to achieve these similarities indicate that ACT has limited

ability to realise cognitive diagnosis. However, several observations from testing the variations of the basic ACT architecture are useful. These are: 1) success in performing an action increases its likelihood of being reused by a student with a new problem, 2) students use models that appear similar to the target (whether the features are relevant or not), and that have a high usage frequency, and 3) analogies based on these factors are very persistent.

2.3.3 Constraint-Based Modelling

The basic premise of Constraint-Based Modelling (CBM) is that not all steps in a problem-solving procedure are equally important. The most important steps are those that change the problem state in some way, particularly those that violate the domain rules (Ohlsson, 1994). To model problem solving in a given domain, the system does not identify every error that a student makes, but instead focuses on determining the instructional actions that the student requires. The analysis that is required to model a domain is to identify solution paths and to capture the main concepts of the domain knowledge as a set of constraints. A constraint has two components - the *relevance* condition (which identifies the problem states where the constraint is relevant) and the *satisfaction* condition (which identifies the relevant states in which the constraint is satisfied). The constraints are then implemented as patterns, and the set of constraints that have been violated by a student constitute the student model. This requires the system to have a means of representing problem states and to have a means of generating solution paths.

This modelling method was tested by applying it to the development of a system for assessing **procedural** skill in the domain of multi-digit subtraction (Ohlsson, 1994), using the problem space that had earlier been used to model the domain (Ohlsson and Langley, 1988). It used a set of constraints to model the domain knowledge, eg. “Increments should not occur in the absence of corresponding decrements (and vice versa)”, and the system was implemented as a production system. The inputs to the system were the problem and the student’s answer. To find a path that explained the

answer, the system typically took between 50 and 100 production cycles. Each state on the path was then compared with the set of all available constraints and the output was the student model, i.e. the set of constraints that had been violated by the student. Thus the system does not attempt to analyse the solution path, to instantiate plans nor to infer the student's goals or strategies. Ohlsson claimed that this is all that is required to identify the instruction that a student needs and that the main advantage of this approach is that it overcomes problems with inconsistencies in problem-solving behaviour (Ohlsson, 1994). However, the current research takes the opposite viewpoint. For problems that require some planning (due to the existence of multiple solution techniques), it is important to identify the one that was chosen by the student and, on a series of observations, to identify inconsistencies in behaviour, ie. the points at which a student changes strategy. The main point of agreement between the current research and that of CBM is that the solution path followed by the student needs to be identified as the key to effective diagnosis.

Sections 2.1 and 2.2 outlined the need for computerised systems that are capable of identifying and explaining the errors that students make when solving algebra problems, and this section detailed a number of approaches that have been applied to the task of cognitive diagnosis in the domain of mathematics. Early success was realised by using rule-based systems to identify errors made by students in the domain of multi-digit subtraction, but this success has not been replicated in the domain of algebra. The reason for this is that problems in subtraction are **procedural**, that is they only have a single valid solution method available and this method is explicit in the problem. However, algebra problems require the student to engage in a number of **conceptual** activities, such as interpreting what the question is asking and planning a solution, not just procedural ones. Systems that approach cognitive diagnosis in algebra from the perspective of the application of a set of skills cannot achieve fine-grained diagnosis unless they can also identify applicable solution techniques. This means that they are limited to diagnosing errors of execution, but are unable to diagnose conceptual errors. Hence, the key issue to be addressed is the identification of the conceptual activities undertaken by the student

when interpreting and solving an algebra problem. In light of this discussion, the next section outlines the results from a number of classroom studies into the learning of algebra. The following section then details several models that have been proposed to describe the processes of human memory and knowledge representation, and the model upon which the current research is based is identified.

2.4 Classroom Algebra Studies

Over the last three decades, there have been many attempts to investigate problems experienced by students in learning mathematics, and to develop classroom teaching materials to alleviate the problems (see Hart, 1981, Davis, 1984, Booth, 1984, MacGregor, 1991 or Swedosh, 1996). In 1974, several members of the Madison Project team in the United States commenced work on creating a new mathematics curriculum (see Davis, 1984). The team members were also involved in teaching and observing student performance. Their observations extended over a five-year period and have become known as the Long Term Study. The results of the Long Term Study (LTS) and other cognitive science approaches to mathematics education can be found in Davis (1984). Two related projects were conducted in the United Kingdom. The first of these was the Concepts in Secondary Mathematics and Science (CSMS) which ran from 1974 to 1979 and the second was a follow-up project called the Strategies and Errors in Secondary Mathematics (SESM) project which ran from 1980 to 1983. The LTS had as its focus the need to identify the conceptual activities required for learning and understanding mathematics (as measured by a student's ability to transfer their existing knowledge to the solution of a novel problem). The UK studies took a narrower focus, viz. identifying the errors that students make and the causes of these errors. The aim of the British projects was two-fold: to investigate problems experienced by secondary students in mathematics and to develop classroom materials to alleviate the problems. The CSMS project identified a number of errors commonly made by students and these then became the focus of the SESM project (see Hart, 1981 or Booth, 1984).

In Australia, the work of Stacey and MacGregor has concentrated on students' understanding of algebraic notation and the cognitive processes involved in algebra. In particular, the authors have concentrated on how students construct and interpret algebraic equations from word problems (MacGregor, 1991, Stacey and MacGregor, 1994, Stacey and MacGregor, 1997a, 1997b). Other Australian studies include those into knowledge acquisition and representation (Sweller, 1989, 1992 , and Sweller and Cooper, 1985) and the hierarchical structure of schematic knowledge (Low and Over, 1992).

A finding common to all of these studies was that rote learning is an inefficient and unreliable way to learn mathematics; instead students need to *understand* individual concepts and the interactions between them before they can progress. The results of these and other studies have been used to generate cognitive models for representing students' mathematical understanding. Several of the major models are discussed in the next section.

We now discuss the findings of some of these studies in terms of knowledge representation, mathematical understanding and problem solving performance.

2.4.1 Long Term Study (US, 1974-1978)

The aim of the Long Term Study was to improve understanding of how students learn algebra in terms of the required information-handling processes. The study involved students in the age group 13-15 years old and used a variety of approaches including observations, think-aloud protocols, interviews and tests. The research team considered all aspects of learning algebra including the representation of knowledge, problem-solving performance and inconsistencies, strategy selection and the impact that teaching strategies have upon the learning process. The research questions that the team addressed included:

- How do students think about mathematical problems?
- Where and why do they encounter special problems?

- What helps to overcome these difficulties?
- What are the natural parameters for learning mathematics, eg. how old should a student be before being introduced to calculus?

The study called upon the results of artificial intelligence to help explain the information processes involved in human learning. The main forms of knowledge representation examined included production systems (rules) and schemata (structures that contain slots that are filled by input data or default values). Davis believed that if a scientific approach is to be successful in education, it must be adopted within the context of an appropriate theory of learning and knowledge representation. He defined mathematics as a collection of ideas and methods built up by the student through identifying and revealing fundamental patterns that often lie hidden beneath the surface. He also noted that:

"*... student errors are not so random as people used to imagine...[they] turn out to be regular and systematic, and it is often possible to predict exactly which wrong answer is most likely to be given by a particular student .... Systematic wrong answers given by a student will often provide valuable clues as to how that student is thinking about a certain class of mathematical problems*."

The results of the study are discussed in full detail in sections six (knowledge representation) and five (errors and their sources) of this chapter.

2.4.2 Concepts in Secondary Mathematics and Science (UK, 1974-1979)

CSMS was the first of two British studies into the processes used by secondary-level students when learning mathematics and science (Hart, 1980, 1981). The project team members set out to determine how difficult secondary school students find particular concepts to be. Data were collected from pen-and-paper tests given to large samples of students from all levels of secondary schools. The tests were generated by the research team in consultation with classroom teachers, educational psychologists and the children themselves.

One of the major findings of the study was that students attempt to extrapolate their methods for solving arithmetic problems to the solution of algebra problems and hence there is a great deal of uniformity in the answers that students give to algebra problems. The most common errors generated on the CSMS tests formed the focus for the follow-up study (Strategies and Errors in Secondary Mathematics). In particular, the CSMS study identified two major points for investigation. The first of these was that the errors made by students in the study were (at least partially) dependent upon the child's interpretation of questions and the use of pronumerals. The second point was that errors made by early learners of algebra arise as a consequence of attempting to extrapolate the techniques used for solving arithmetic problems to algebra. Both error types are examples of what we refer to as *conceptual* errors, which are discussed in more detail in section 5 of this chapter.

2.4.3 Strategies and Errors in Secondary Mathematics (UK, 1980-1983)

The SESM study was based on a cohort of students from Years 2-4 of secondary schools in the United Kingdom. Lengthy interviews were conducted with 27 children in the 13-15 age group, and the final tests were given to 3350 children in the same age group. The aim of the research was to explore the effectiveness of short teaching modules, which had been designed to help children to correct or avoid errors in elementary algebra. It focused on the use of pronumerals in writing general statements to represent given arithmetical rules and operations, but did not consider factorisation, the solution of equations or simplification of complex expressions.

The study revealed that teaching targeted at remediating problems was successful (if limited). Particular areas of difficulty experienced by students included the interpretation of letters, formalisation of methods, and understanding of notation and convention (eg. students have an initial preference for working left to right rather than employing precedence of operations and the use of brackets). The researchers concluded that these difficulties may be rooted in arithmetic rather than algebra *per se*, because children do not

have the appropriate structures for arithmetic that can be generalised to the case of algebra. Cognitive growth occurs when the student assimilates concepts and procedures, which in turn requires the development of particular cognitive structures that take time and maturation (*cf.* Piaget). These findings are in line with the "back and forward" model of mathematical understanding proposed by Pirie and Kieren (Pirie and Kieren, 1994). The researchers also observed similarities in the nature of the informal methods used by students and concluded that this suggests that they have a root in some generality in cognition (this idea is explored in greater detail in sections 4 and 5 of this chapter). The quote below is particularly relevant to the current research.

"*The research ... suggests that a fuller understanding of algebra ... requires that attention be paid to the kind of methods that children use, and to ways of assisting children to become aware of the uses and limitations of different kinds of procedure. Attention is also required to the ways in which these procedures can be symbolised. As a first step, this requires the teacher to recognise that the child may be using informal methods rather than the taught procedures, and to investigate the kinds of informal method which are so used, drawing the child's attention to the nature of these methods, their usefulness and their limitations. Only when children become aware of the limitations of their own methods, it is suggested will they be prepared to contemplate the value of the more formal methods which the teacher is attempting to teach. This focus on the nature of the procedure required to solve a given class of problems may also help to shift children's attention from an over-concern with the final (numerical) answer.*" (Booth, 1984, p. 93)

These classroom studies and other research into problem-solving expertise have led to theories of how students approach problem solving in mathematics and hence of the error-making process. The next section provides details of the relevant research findings, which form the basis of the model of algebraic problem solving developed in chapter three.

2.5 Algebraic Problem Solving and Errors

In the current research, we are attempting to improve cognitive diagnosis in the domain of algebra. Algebra has received a great deal of attention from both classroom specialists and the interactive learning environment fraternities, because it is the basis upon which most mathematics and scientific problem-solving are based. It is also true that algebra is the first level of abstraction encountered during mathematical learning.

There is broad agreement between cognitive scientists about the manner in which students approach mathematical problem solving (see chapter 3 for full detail). The basic notion is that students employ some type of analogical reasoning to effect transfer of their knowledge by matching a familiar problem stored in memory (the base) to the new (or target) problem (Kimball, 1982, Davis, 1984, Anderson, 1990 or English, 1997). The base problem is then adapted in terms of the new problem. To achieve analogical transfer, the student must have an internal representation of the original problem that enables recognition and retrieval to be effected. The similarity of (or correspondence between) the base and target problems must be noticed and the appropriate base problem retrieved. A mapping between the base and target problems is then created and the activities associated with the base problem are then carried out. The most important aspect that impinges on the success of analogical transfer is the student's knowledge representation, because the notion of similarity is reflected by the content and structure of the knowledge base. The major difference between arithmetic and algebra is that, to solve an algebra problem, the student must plan a solution strategy (since, unlike the case of arithmetic, the operation(s) required for solution are not always explicit) and then execute the strategy. The choice of strategy is affected by what the student believes will effect the desired change from the initial state to the goal state.

To generate a plan for solving a problem, the student must firstly interpret the question (i.e. determine what it is asking) and then choose an appropriate solution technique. This

will either be a complete plan (if the student has a complete view of the problem) or a partial plan (if the student cannot classify the problem at the finest level of granularity and needs to adopt a means-end approach to solving the problem). In the case of a partial plan, the student may adopt an appropriate but suboptimal solution technique, which is considered to be a conceptual error because the student needs to expand their mental models to accommodate specialised techniques and the criteria for applying these. The aim is to improve their view of a problem, and to look beyond surface features.

Throughout the literature, we find discussion of individual errors that are commonly made by algebra students. There is a great diversity in the errors that students make and creating a complete list of them may not be practical (Davis, 1984). Davis explained errors in terms of the learner's knowledge representation structures and the procedures the learner has developed to solve multi-step problems. In an attempt to determine the cause of the common errors, Davis drew a distinction between concepts and skills and the types of errors related to each. A "skill" is the ability to accurately execute a sequence of actions. A "concept" is a mental structure which enables a student to classify a problem and plan a solution. Students can often perform a task once they are told what to do (i.e. which solution technique to apply) but there are conceptual hurdles in determining which procedures to perform, due to insufficient or inaccurate pattern recognition. Problem solving in algebra exposes the student to two major sources of error, viz. **conceptual** errors (those made during the planning stage) and **executive** errors (those made whilst executing the solution strategy). Executive errors have one of two basic causes: errors made in performing a single step in the solution plan and errors in control (those which indicate that the student has either prematurely stopped or has otherwise lost control of the solution process). In this section, we examine the major error types and the causes that underpin them.

2.5.1 Conceptual errors

Conceptual errors are errors made when planning a solution strategy. They arise for different reasons but are functionally dependent upon the student's knowledge base, and

interpretation of the question. Planning a solution requires the student to have some knowledge of the form that the desired result should take. To achieve the goal state, the student may adopt one of several methods:

- *means-end analysis* - creating a partial plan at the outset of a problem, executing it and then reviewing their progress (the student iterates through these steps, attempting to reduce the difference between the current problem state and the desired end state, until they believe that they have achieved their goal),
- *constraint propagation* - occurs when the student has an idea about the form that the final answer should take and uses this knowledge to guide the solution process eg. when factorising a quadratic expression the first step could be to write ()*() and then to use heuristics to determine the terms to write inside each of the sets of brackets, and
- *back propagation* - attempting to work backwards from the goal state to the initial state when there is no obvious heuristic to guide a forward search. This requires that the student be able to identify the goal state and possible intermediate problem states, and is commonly applied to problems that require the student to prove a proposition or to find a path between two states (eg. network problems).

Within the domain of algebra, the goal of the problem is the best pointer to the form that the answer should take, but identifying the goal can be difficult, particularly when the keyword has some subjective element to it, an example being the keyword *Simplify*, which can be interpreted in different ways depending upon contextual information. In the case of the algebraic fraction in Figure 2.4, some people regard the form on the left-hand side to be the simpler form of the two, whereas the factorised form is more likely to be useful (particularly if the fraction is embedded in some larger problem).

$$\frac{x^2+4x+4}{x^2+5x-6} = \frac{(x+2)^2}{(x+6)(x-1)}$$

Figure 2-4 Simplifying an algebraic fraction

Even for algebra problems with a keyword that is unambiguous (such as *Factorise*), students may not fully appreciate what is required of them. Less able students are aware that factorisation implies something about brackets and therefore believe that the working in Figure 2-5 satisfies the goal.

$$\begin{aligned} &\textit{Factorise} \quad (x+3y)^2 - y^2 \\ &= x^2 + 6xy + 9y^2 - y^2 \\ &= x^2 + 6xy + 8y^2 \\ &= x(x+6y) + 8(y^2) \end{aligned}$$

Figure 2-5 Misconception of the term "Factorise"

This working indicates that the student has facility in executing skills (such as expanding a perfect square or collecting like terms), but does not really understand what the keyword means, i.e., their concept of "*Factorise*" is not fully developed.

Davis noted that "nearly all procedures used by a student were correct for some earlier task, as performed in some earlier context" meaning that the student has either applied the right rule in the wrong place or has adapted a correct rule in an inappropriate way (Davis, 1984). The remainder of this section details some of the common conceptual errors.

2.5.1.1 Binary Reversions

Errors in problem-solving occur when the learner either retrieves an inappropriate schema or has incorrectly adapted an existing schema (see section 2.6 for detail). Davis defined "Binary Reversions" as the situation that occurs when a student answers a question (Q2), which is similar to the question that was asked (Q1). Typically Q2 is of easier type than Q1 and has been met by the student earlier than has Q1. Also, the visual cues for the two questions are often very similar and hence have similar schemata. Greater familiarity with the Q2 schema leads to its retrieval instead of that for Q1. For example, in arithmetic, when a learner first encounters a problem such as **4 x 4 = ?** , it is common for the answer **8** to be given, because the visual cues for addition and multiplication are very similar and addition is typically the first mathematical operation that students are taught.

This type of error is encountered in other forms, for example students will often apply differentiation techniques to the solution of problems in anti-differentiation. For example, it is very common for students to produce the answer $\int \ln x.dx \to \frac{1}{x}$. In this example, the student has differentiated the integrand, which may be a binary reversion introduced by a lack of understanding of the technique of Integration by Parts. Related to this is another persistent error that occurs when a super-procedure selects the wrong sub-procedure (evidenced by the student making the same mistake at the same place each time). Retrieved schemata tend to be very persistent to the point where a student who began with the correct equation will change their definition of variables etc. to accommodate the old, incorrect schema. Students who do not tend to check their results will often retrieve the wrong schema but do not recognise this retrieval error.

2.5.1.2 Extrapolation Errors

Matz was one of the first researchers to attempt to model errors from the point of view of forming a computational model (Matz, 1980, 1982). In her work, she accounted for some of the errors that students make at the planning stage; these errors can also accounted for by Davis's model of mathematical learning as they provide evidence for the modification of existing schemata and the construction of new ones. Matz identified a family of errors that arise from a student's extrapolation techniques, i.e. the methods used to extend or generalise known techniques to a novel problem (Matz, 1980, 1982). She listed a number of common algebra errors that are covered by the theory, which is based upon two assumptions: that there are limited resources for problem solving (short-term memory and the student's knowledge base) and that these resources are used in a rational way. Supporting evidence for the theory is provided by the similarity of errors made by students from different backgrounds, who have received different teaching methods and have different learning styles. These errors are now detailed.

2.5.1.2.1 The Null Factor Law

The Null Factor Law states that if, $x.y = 0$ then either $x = 0$ or $y = 0$. It is common for students to extend this law to $x.y = 3 \Rightarrow x = 3$ or $y = 3$. This indicates that the student does not understand the significance of the zero on the right-hand side of the equation.

2.5.1.2.2 Generalised Distribution

Another common extrapolation is to assume that all operators are linear. The name arises from the relationship between this erroneous technique and the Distributive Law. The Distributive Law is encountered very early in the studying of mathematics and is very powerful. It is possible that the source of the error is aural "do the same thing to each term inside the brackets". This may not be recommended (Davis *et al*, 1978), but teachers **do** say it. This error type is persistent and ubiquitous, but can usually be remediated by numerical substitution. Misgeneralisations of the Distributive Law occur in all areas of mathematics including algebra (eg. $(x+y)^3 \rightarrow x^3 + y^3$), in trigonometry (eg. $\sin(a+b) \rightarrow \sin a + \sin b$) and calculus (eg. $\frac{d}{dx}(2x.e^x) \rightarrow \frac{d}{dx}(2x).\frac{d}{dx}(e^x) = 2.e^x$).

2.5.1.2.3 Repeated Application Errors

Repeated application errors occur when a student extends a rule from a prototype to a problem with extra terms. These are particularly common when dealing with algebraic fractions and cancellation.

Example 1. $\frac{1}{3} = \frac{1}{x} \Rightarrow x = 3$ is extrapolated to the case $\frac{1}{3} + \frac{1}{2} = \frac{1}{x} \Rightarrow x = 3 + 2 = 5$.

A similar error occurs when students attempt to cancel additive terms in an algebraic fraction.

Example 2. $\frac{ax + by}{x + y} \rightarrow a + b$

Example 3. $\frac{\left(x^2 + 2xy + y^2\right)}{\left(x^2 + y^2\right)} \rightarrow 2xy$

Some solution techniques are especially prone to extrapolation errors. An example of such a technique is the method for expanding the product of two binomial terms called FOIL (first, outside, inside, last). This technique (which is just the repeated application of the distributive law) will always produce the correct answer when applied to the product of two binomial terms (excepting arithmetic errors), but if students attempt to apply it other situations (such as expanding a cube), the method fails. Despite the fact that this appears in the majority of text-books, it should be avoided.

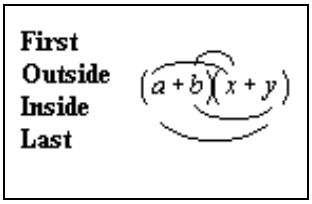

Figure 2-6 The FOIL technique

If a student reaches an impasse when solving a problem, it is rare for them to do nothing. Instead, they will attempt to create a solution technique called a "repair". Most repairs are not replicated; that is, once they have served their purpose they are no longer required and so they are not stored (van Lehn, 1983). This sort of performance can be affected by the student's level of confidence. For example, consider the problem *Simplify* $\sqrt{a^2 - b^2}$. Most students will produce the answer "$a - b$" (see chapter 4). Part of the reason for this is that in the classroom, students are rarely exposed to questions that cannot be further reduced or which have excess information. In the case of irrelevant information, students will typically attempt to incorporate all information into the solution process, and if there is insufficient information students will attempt to construct their own solution method based on the information present (Low and Over, 1992). One student who was faced with this problem produced the answer "$\sqrt{a^2 - b^2} = c$, by Pythagoras" (see chapter four). This is an example of rote learning in the absence of understanding - the "method" is completely irrelevant for the given problem and indicates that the student has failed to correctly interpret the question.

Assuming that students are methodical in the application of their knowledge, then remediating conceptual errors becomes a matter of identifying what they know and deem relevant to a particular problem, and what rules govern the application of their knowledge. Apart from the skills required for executing the steps in a solution plan, students must also acquire supplementary knowledge about the “rules” of algebra, such as knowing why a particular rule or method works and which features of a rule's pattern are fundamental to its success and which are incidental (eg. the significance of the zero in the Null Factor Law). During problem decomposition students need to know which sub-problems are legal (eg. algebraic fractions). Finally, they need to be encouraged to anticipate and verify the results of a procedure (this aids in the creation of critics which are discussed in section 2.6 of this thesis).

2.5.2 Executive errors

We now consider two types of errors that can occur during the execution of a solution plan. These are procedural errors (those made when performing a single step of the solution plan) and control errors (those that indicate that the solver has either performed steps out of order or has omitted steps from the solution plan).

2.5.2.1 Procedural errors

Processing errors are made inconsistently (and are context dependent) and often self-corrected. Consider the example below (what Matz referred to as "The Lost Common Denominator"). Even competent problem solvers can make this error, particularly when it is part of a larger problem. This is an error in the general procedure of cross-multiplying fractions to normalise the denominator

$$\frac{5}{(2+x)}+\frac{5}{(2-x)}=4 \quad \rightarrow \quad 5(2-x)\ +\ 5(2+x)\ =\ 4$$

Procedural errors have been well documented (for example, Davis, 1984, Matz, 1980, or Sleeman, 1984). These are used as the basis of the bug library in various diagnostic systems (see the next section for more detail).

2.5.2.2 Control Errors

Davis drew an analogy between human memory and computer memory. He believed that human memory is partitioned into a large area called "passive memory" (which is like a storehouse of facts and procedures) and a smaller area called "work-space memory". Work-space memory is further subdivided into three sections: control memory (which governs the procedural aspects of problem-solving), abstract memory (which contains tokens to represent the details of a problem) and whole-text memory (which contains full details of a few items). When solving a problem, we retrieve relevant facts and procedures from passive memory and copy them into work-space memory. During the solution of a problem, different items held in work-space memory will be overwritten as other items need to be brought into memory.

Davis believed that the human working memory operates like stacks, which are typically used by computers and calculators during complex calculations. During problem solving, the order of the stack may need to be reorganised. Problems occur if the items in the stacks are in the wrong order and finding the correct order may not be easy. There is also a maximum length of stacks for humans, which is quite small. One suggested cause of errors is that students may lose items off the top of the stack (Matz, 1980). For example, consider the following application of the Distributive Law: $x(y + z) = xy + z$. This error has a number of alternative explanations. Firstly, it may indicate that the student has made a control error (ie. has lost track of all the required sub-steps). Secondly, it could arise as a result of either missing skills or misconceptions. Thirdly, the error may have an auditory explanation; that is, the student reads the expression as "x times y plus z". This explanation also accounts for a common error made when collecting like terms, viz. $2y - y \rightarrow 2$, which is read as "Two y take y", yielding the resultant answer.

Another example of a control error is when a student stops the problem-solving process prematurely. These are usually associated with the means-end approach to solution planning, particularly when each subproblem has a different goal. Consider the factorisation example in Figure 2-7.

Factorise $(x+3y)^2 - y^2$

$(x+3y)^2 - y^2$

$= x^2 + 6xy + 9y^2 - y^2$

$= x^2 + 6xy + 8y^2$

Figure 2-7 Means-end approach to factorisation

This is a common response to the problem. By changing goal from Factorise to Expand, the student can lose track of the process. Once the bracketed term is expanded and the like terms have been collected, the student stops. Alternatively, the student may perceive that they have moved further form the original goal and perceive the local failure as a global failure of the procedure to produce the desired result. Rather than looking for another approach, the student quits.

One final type of control error is when the steps in the solution plan are executed out of order. For example, the error $2(x+y)^2 \rightarrow (2x+2y)^2$ arises when the correct precedence of operations is not observed.

Before concluding, we consider one final error type, viz. the errors that students make when *checking* their solutions.

2.5.3 Checking errors

Davis noted that most of the procedures that students apply to problem solving were successful in some former experience, and the students attempt to reuse the plan by adapting it to the current problem. If the student lacks appropriate knowledge structures, then they have no method for checking their results. Alternatively, they may create

erroneous checking rules that are in line with the assumptions that they made when developing the technique, which would help to explain the persistence of errors such as linearity errors. The types of errors that students make when solving linear equations in a single unknown have been identified in the literature (eg. Sleeman, 1984 and Payne and Squibb, 1990). Some of these errors can be detected and corrected by the student themselves, but this requires the student using correct checking procedures (Perrenet and Wolters, 1994).

Checking errors, like problem-solving errors, have different causes and can also be divided into structural (or conceptual) errors and executive errors. An example of a structural checking error is shown in Figure 2.8. In this case, the student is aware that they want the two sides of the equation to be the same when the substitution is effected. To ensure that they do obtain an identity, they change the equation into which the found value is substituted. This indicates that there is a basic flaw in the student's understanding of the concept of checking. Perrenet and Wolters found that these types of checking errors were associated with students who scored very poorly on tests (i.e. those who made the greatest errors in problem solving). The errors were either logic-based (that is, there is a flaw in the logic of the checking procedure) or operator-based (where the student breaks some of the rules of algebra, such as assuming that subtraction is commutative).

Solving	*Checking*
$7-\frac{1}{2}n=18$	$11+7=18$
$\Rightarrow \frac{1}{2}n=11$	*OK*!
$\Rightarrow n=22$	

Figure 2-8 An erroneous checking rule

The second type of checking error (executive) was associated with students who scored well in problem solving. These errors were usually self-diagnosed and self-corrected. An example is shown in Figure 2-9. In this example, the student wrote down very few steps (either during problem-solving or checking), and when asked to step through the checking procedure, he self-corrected both his checking and problem solving. Perrenet and Wolters

ascribed many of these executive errors to a combination of a lack of care and attention, laziness and too much self-confidence. Typically, some steps of the checking procedure are omitted by these students, whereas students who make structurally erroneous checking procedures do so from a lack of understanding of the processes of algebra.

Solving	*Checking*
$7-\frac{1}{2}n=18$	$7-\left(\frac{1}{2}.-50\right)=18$
$\Rightarrow \frac{1}{2}n=-25$	*OK*!
$\Rightarrow n=-50$	

Figure 2-9 A second erroneous checking rule

The message that we receive from this is that, whilst reflection on results is an important part of the process of learning mathematics, it too can be error-prone. Hence, checking answers is a skill that students need to practise.

Throughout this discussion, reference has been made to a student's knowledge structures. The next section reviews two theories of human memory and knowledge representation relevant to the learning of algebra, viz. schema theory and production systems (or rules).

2.6 Human Memory and Knowledge Representation

Recently, my sister was making up her grocery shopping list and she wanted to include baked beans, however she could not recall the full name of the item. The following conversation ensued:

Sister: What is the name of the beans that come in cans?

Self: Fava beans? Kidney beans?

Sister: No, no, no ... you know, Tim Brooke-Taylor in his school-boy uniform...

Self: Ah - *baked* beans.

Sister: **Baked** beans - thank you!

In this episode, the first person wanted a type of beans and could recall that they came in a can. However, she also recalled in detail a video-clip from the BBC television show “The Goodies”, which in turn served as a rich stimulus to a second person (who had the same image encoded in memory) and could then recall the full name of the desired object. Both people had a detailed representation of the object “Baked Beans”, which included a visual image that could itself be used (in conjunction with the verbal cue "beans in a can") to index memory and prompt retrieval.

This type of performance points to a form of knowledge representation that is richer than purely verbal encoding and is typical of how our long-term memory works, in that we encode both generic examples (or stereotypes) as well as specific episodes (see Schank and Kolodner, 1979). The same is true of both teachers and learners of mathematics (English, 1997). For example, the term “quadratic equation” may bring to mind a visualisation of a parabola, the quadratic formula for determining the roots of an equation or a particular application such as projectile motion. We encode memories in a variety of formats and the stronger the links are between the different memory fragments, the more useful they will be in the problem-solving process.

Whilst cognitive scientists have proposed a great variety of models of human memory and information processing, there is general agreement between them about how memory operates (see Simon and Kaplan, 1989 for a description of the basic models). The consensus is that memory is divided into two sections: short-term (or working) memory and long-term memory. The two components have different tasks and operate in very different ways. Long-term memory is the storehouse of knowledge (facts, procedures etc.) that we develop during learning. It has virtually unlimited capacity and hence access times can be slow. Storage, accommodation and indexing of new information can take in the order of 8-10 seconds to achieve. Retrieval of a block of information (a "chunk") during problem solving can take 1-2 seconds, but subsequent related chunks are retrieved much faster (usually in the order of hundreds of milliseconds) because of the existence of

pointers/indexes that indicate that we are in the general area. Another important aspect of long-term memory is that it has an associative structure. Short-term memory on the other hand is very limited in its capacity (about 7 chunks) and is very volatile, but access is exceedingly fast (in the order of hundreds of milliseconds). The main purpose of short-term memory is to hold the most immediate items required for problem solving.

Two models that have been proposed to account for human knowledge representation are schemata and production systems (or rules). We now compare these two theories and outline the one upon which our work is based. The methods that can be used to implement knowledge representation in a computerised system are related to the methods that human use and so two main methods for implementing automated reasoning (viz. rule-based and case-based reasoning) are discussed in the following section, and a justification is given for the choice of paradigm used to implement the current research.

2.6.1 Schemata and Critics

Davis summarised problem solving as a process involving two parallel activities, viz. a *constructive* phase (which involves building representations, mappings etc.) and an *analytical* phase (which uses meta-language to guide the construction processes). He summarised problem solving as a 10-step process as follows:

- visual cues to trigger retrieval,
- retrieval,
- build up mental representation of the problem,
- build up mental representation of the related knowledge,
- map inputs to problem representation,
- map problem representation to knowledge representation,
- evaluate the adequacy of retrieval and mappings,
- use heuristics and set subgoals,
- cycle back and forward, and
- invoke a stopping rule.

Based upon the results of the Long Term Study, Davis proposed a cognitive model where human memory is partitioned into a large storage area and a small working area. It is the work-space memory that is overwritten during problem solving. To make use of existing knowledge requires locating it in passive memory and copying it into active (or working) memory. The types of knowledge encoded in passive memory include facts (eg. 7+8 = 15), procedures (sets of instructions for completing a particular task) plus meta-knowledge of the domain. When solving a problem, a student needs to access all the relevant knowledge and so the knowledge needs to be indexed in a manner which facilitates complete and accurate recall. Because a procedure can be relevant to a number of different tasks, its representation in memory will contain slots for variables which will be instantiated as required.

In multi-step problems, a sequence of procedures is required for solving. Because of the limitation of space in working memory, procedures are only recalled as required. Davis postulated that solving complex problems is a means-end task where a combination of visual cues (what he termed “visually moderated sequences”) and the current problem state will determine which procedures to recall. In summary, the problem solving process has three basic steps: analyse the input for cues, categorise the question by matching it with stored knowledge representation structures and to execute the associated solution technique. In the case where the student has no complete procedure encoded for a particular problem, Davis believed that the student will perform some type of real-time synthesis to create the required procedure based upon some common elements that must be retrieved from storage. This means that the student recalls knowledge relevant to the current problem and adapts it to the new situation.

Early learners tend to encode individual procedures independently of one another, whereas experts form links between the various “chunks” of knowledge and can encode super-procedures (integrated sequences which comprise a range of techniques) for complex tasks. Hence, as they learn, students expand and adapt their existing knowledge

to accommodate new elements by forming abstractions of related topics and links between these. By forming such associations, the student improves their efficiency in problem solving. This is because the recall process can be improved in a variety of ways, viz. simultaneous loading of related structures, branching from one procedure to another depending upon the values of parameters, and adjusting parameters (Davis included a parameter called "urgency" which is used to determine priorities for various structures).

For a given problem, its degree of difficulty will vary from student to student and is a function of both the problem representation and the student's own knowledge representation. Just how the various types of knowledge are encoded in human memory is problematic. Indeed, this is the major difference between the two models being discussed here. Davis described a range of human knowledge representation structures including production systems (sets of IF-THEN rules). However, because he believed that students use a variety of mental representations (including verbal, graphical and hybrid forms), Davis proposed two major types of cognitive constructs in passive memory, which he called frames (or schemata) and critics (Davis, 1984).

The two types of structures have quite different forms and functions. A *schema* is one type of construct used for storing information about the structure of a problem and the steps that can be followed to solve it (an example is shown in Figure 2-10). In this way, a schema is like a template which contains slots to be filled either by input values or default values (Schank and Kolodner, 1979). A *critic* is a different type of knowledge structure that enables a person to create analogies and interpretations, to determine how reasonable an answer is, and that helps to control the problem-solving process. Novices will have few of these, whereas an expert may have thousands. These may not fire reasonably or consistently for novices. This does not generate an error but allows errors to persist. The absence of such critics leads the novice either to incorrectly adapt a known rule to fit the new problem or to fail to understand the significance of particular numbers (eg. the zero in $x^2 - 5x - 6 = 0$). To encourage learners to create new schemata and accompanying

critics, the teacher needs to emphasise validation and justification as important aspects of the solution process.

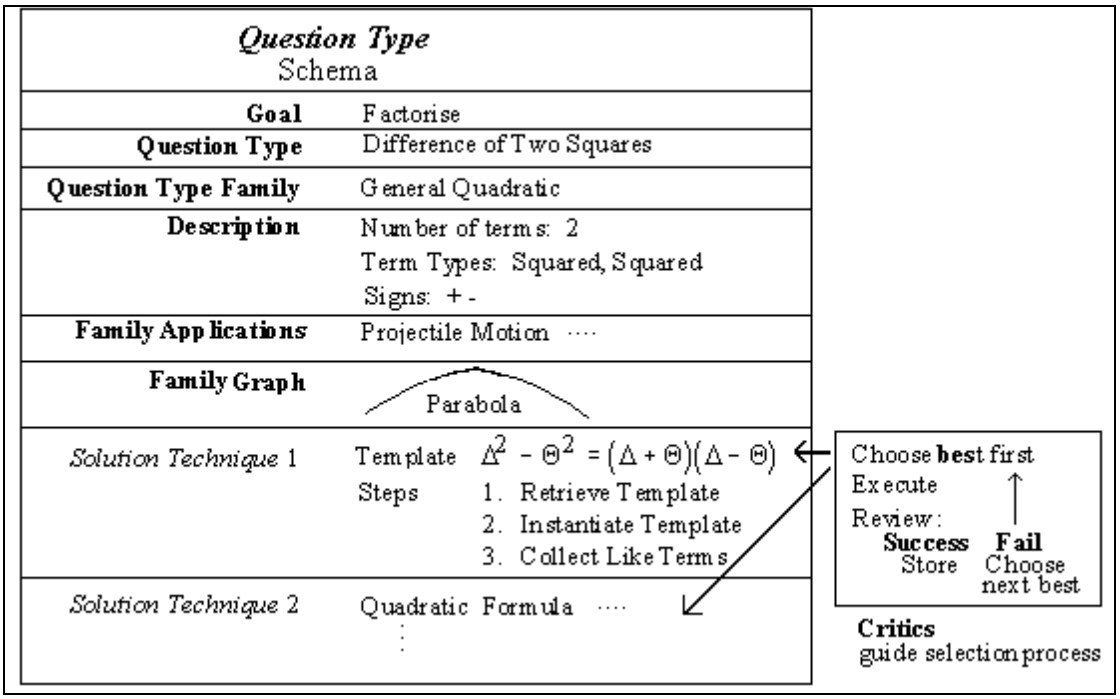

Figure 2-10 An example of a schema

The mechanism for dealing with a novel situation involves retrieving an initial interpretive schema (based on some cues in the input data), retrieving the prompted schema (and automatically see a list of previous ones) and, if there is overlap between previous solutions and this case, selecting the relevant pieces. Thus a schema acts as a method for assimilating and organising input data. Each schema has its own correct context (i.e. problems where it produces correct answers), and requires certain input information. Schemata are also persistent (whether appropriate or not). Davis claimed that errors in problem-solving occur when the learner either retrieves an inappropriate schema or has incorrectly adapted an existing frame. Problem solving also requires some control structure to oversee the solution process and so Davis postulated the existence of control memory which has the purpose of keeping track of what has already been done, what still remains to be done as well as where the partial or interim results are stored (see section 2.5 for more detail).

There are a number of issues associated with the use of schemata as a means of representing knowledge. These include knowing when to terminate the search for relevant schemata (particularly in the case of no success), how to transform a given problem to something familiar (eg. $e^{2t} + 5e^{t} + 6 \Leftrightarrow x^{2} + 5x + 6$), how to map input data to schema variables, determining default evaluations (if no suitable input is available), judging the appropriateness of different schemata in a particular situation, schema adaptation (usually entails adding the new one without deleting the old one), and reflection on results. Pattern recognition and combining cues are ways of reducing the memory search problem. These issues are addressed in chapter 3.

2.6.2 Production Systems

In this section, we review two cognitive architectures that have been encoded using production systems, but which vary in their knowledge representations of long-term memory. These architectures are the SOAR architecture (Laird, Rosenbloom and Newell, 1986 and Laird, Newell and Rosenbloom, 1987) and the ACT architecture and its modifications (Anderson 1990, 1993b, Anderson *et al*, 1997, Anderson and Lebière, 1998, Anderson, 2000a, 2000b).

2.6.2.1 SOAR Architecture

Newell *et al* pointed out that it is possible to replicate the behaviour of an artefact in an infinite number of ways. This insight led them to develop “principled simulations” and to build architectures which incorporate these principles including the SOAR architecture, which is based upon a general model of problem solving (Newell and Simon, 1972). SOAR was proposed as an architecture for intelligent problem solving and learning (Laird, Newell and Rosenbloom, 1987). It has many similarities with ACT theory, so in this section, we focus on the differences between the two architectures.

In SOAR, short-term memory includes information about goals and preferences. Because it is linked to goals, it is volatile - once a goal is realised, the associated information is

overwritten. Long-term memory is a single, associative entity; that is, both declarative and procedural knowledge are encoded as productions. Objects in long-term memory conform to a uniform scheme of sets of attributes and values. Problem solving becomes a task of determining which problem states are desirable, and which operators can be used to achieve these. The choice of problem states and operators is guided by preferences that have been copied from long-term memory into working memory (these can be compared with the *critics* proposed by Davis). The selection phase is a two-stage process. The first step, called *elaboration*, is to repeatedly access long-term memory. Rather than needing to resolve conflicts when multiple productions are vying for activation, SOAR fires all contending productions in parallel. Once all available information has been collected, the system makes a selection that is based upon the preferences. By examining multiple problem spaces, SOAR has a mechanism for dealing with impasses in problem solving. In the absence of the full knowledge required for the decision-making process, the architecture generates subgoals and solves the problem recursively. This information leads to the creation of new productions; which is called “chunking”.

2.6.2.2 ACT Architecture

ACT theory (like most theories of cognition) postulates that human memory is divided into two main components: long-term and short-term memory. Long-term memory is further subdivided into a *declarative* portion (representing the store of facts and concepts) and a *procedural* portion (representing the operators required for problem solving). In ACT, declarative memory is represented as a semantic net (with objects and concepts as nodes and the relationships between these as arcs) whilst procedural memory is encoded as rules. Elements in both types of memory are associated with an activation pattern, which is functionally dependent upon the usage of the associated knowledge. Short-term (or working) memory is used to hold a small number of items during problem solving. Since there is a limit to the amount of information which can be held locally in working space memory, the individual requires access and retrieval functions to link to other knowledge. Changes occur in working memory only when some activation functions

change. Anderson used Bayesian statistical analysis to compute probabilities for the activation patterns that determine which rule is fired.

Other factors which affect a human's processing include recency, frequency, success and context. That is, a problem-solving strategy which has been successful in the past is more likely to be applied to a particular problem than one which had only partial success. It is also the case that the more often and the more recent this success, the more likely it is that the individual will attempt to reuse or adapt the technique. These aspects will determine the activation pattern for a given solution technique. Persistence of techniques in problem solving is a function of their transferability, frequency, recency and success in the past. Another aspect which influences the choice of a solution technique is the amount of effort required to solve the problem using that technique. That is, students will attempt to adopt a simple technique before one which involves more computation or mental exertion. Context may provide cues for which technique to adopt - that is, cues change the activation patterns.

Anderson noted the need for functions which can access and retrieve items from memory. These functions require the classification or categorisation of items. Categorisation is based on the notion that if objects share common features or attributes then they can be grouped into the one category, and the categories can be hierarchical. Thus, if we can determine an object's category then we are in a position to predict much of the object's behaviour. The better we can predict an object's behaviour the better will be the adaptation of our behaviour. Categorisation requires that we can measure how similar objects are but it is not an end in itself - rather its real purpose is for prediction. When a new object does not match an existing category, a new category node must be created.

The final component of the ACT-R theory is *causal inference*. Anderson contends that humans use causal inference as a means of optimising predictions and that such inference is represented as rules in memory. However it is possible under ACT-R theory that multiple rules will fire in the one circumstance. This is contrary to the general

assumptions made in production rule architecture and suggests that a less deterministic representation would be better. For example, matching a current problem to an existing one in memory may lead to the retrieval of several competing possibilities. The person then needs a means of measuring the similarity between the contenders and of choosing the most likely candidate. Typically similarity will be a function of the surface features of the problem.

In summary, there is a great deal of similarity between the theories of Anderson, Newell and Davis - the main difference being the manner in which human memories are encoded. Whilst ACT and SOAR can reproduce human behaviours, they do not represent what humans actually do. For example, ACT makes extensive use of utility and probability in its optimisation functions. Two main criticisms of ACT theory are: 1) it seems more likely that humans use frequencies rather than probabilities to determine which knowledge structure to apply, and 2) it is generally accepted that standard utility theory does not offer a good account of human decision making (Simon and Kaplan, 1989). A related issue is the lack of evidence to support the idea that humans generate productions.

This section has outlined two models of human knowledge representation - schemata and rules. The current research is based upon the former of these models and hence a decision had to be made about how such knowledge could be represented in a computerised system. Therefore the next section reviews two paradigms for implementing reasoning in a computerised system. These are rule-based reasoning (which has been used to implement a large number of expert systems in a variety of domains) and case-based reasoning (which is closely aligned with the general approach to problem solving outlined in chapter three).

2.7 Automated Reasoning and Cognitive Diagnosis

Reasoning in a computer can be considered to comprise several activities: organising and analysing facts and data, associating a set of inferences, drawing hypotheses/conclusions etc. For software to be considered intelligent, all of these activities should be included and carried out when required. Because we aim to capture the problem solver's cognitive processes, cognitive diagnosis is a different (more complex) task than, say, fault diagnosis in machinery (see Self, 1996 and Patterson and Hughes, 1997). In the latter case, the system is built around a model of an expert problem-solver (the mechanic) working with a deterministic machine, whereas the current research aims to model learners of algebra whose performance cannot be expected to be a stable process. As a student acquires greater knowledge and makes the transition from novice to expert, their problem solving behaviour varies, but this cannot be assumed to be uni-directional because students can forget as well as learn. Hence, we cannot assume that because a student has shown mastery of a particular concept in one question that they will subsequently always demonstrate the same ability. Rather, problem-solving behaviour is influenced by contextual information and can regress under cognitive load (see Brown and van Lehn, 1980, Sleeman, 1984, Sweller, 1992 and Lebière *et al*, 1994). In this section, we provide an overview of two types of automated reasoning and how they can be applied to cognitive diagnosis.

2.7.1 Rule-based Reasoning

One of the earliest methods of reasoning in Artificial Intelligence (AI) was Rule-based reasoning (RBR). This method has gained a great deal of support in the AI community, particularly for the production of expert systems, and a number of languages have been developed for creating expert shells, including PROLOG and LISP. A rule-based reasoner (or production system) typically contains three main components: a set of rules, a control structure to guide the reasoning process and working memory.

Rules have the general form IF {*conditions*} THEN {*actions*}. Working memory contains information about the current state of the problem solving process. Inputs may come from either the user or the reasoning that the system has already undertaken stored as an *object-*

attribute-value triple. A rule is fired when the contents of working memory match either the condition set for that rule or the action set for the rule. In the first type of match, goal-driven reasoning or *backward chaining* occurs. In summary, this means that if the problem can be stated as prove X where X has the form $A \rightarrow C$ and we can find a theorem which proves that $B \rightarrow C$, then the original problem can be transformed into the equivalent form of proving that $A \rightarrow B$. This involves determining the final goal and the rules and conditions are needed to achieve this goal. These conditions become the new goals. The process continues until a path is found from the original state to the desired target. In the second type of match, data-driven reasoning or *forward chaining* occurs. The original problem can then be restated as prove $B \rightarrow C$, where $A \rightarrow B$. Searching begins from the original state and applying all valid rules to generate a new set of facts. This continues until a path from the original state to the target state is found. Both types of search yield exponential complexity.

The control structure has the task of determining which rule to fire during a machine cycle. If there is more than one candidate rule, a set (called the *conflict set*) is formed and it is the task of the control structure to resolve the conflict and determine which rule to fire first. Strategies used for conflict resolution include *textual order* (firing the rules in the order in which they appear in the rule base), *refractoriness* (the same rule should not be applied more than once to the same data), *recency* (firing the rule which was most recently used) and *specificity* (the rule which has the highest number of highest number of instantiated variables is fired first).

Advantages that have been claimed for rule-based systems include a natural means of expressing experts' problem-solving heuristics, modularity of knowledge organisation, restricted syntax simplifies the programming task, frequent examination of working memory can help to direct the problem-solving process and explanation for a solution can be facilitated by retracing the rules that were used. Many of these properties that have been claimed as advantages of rule-based systems are also cited as disadvantages. In particular, **explanation** for the reasoning process has been criticised because it omits

knowledge about the diagnostic approach used by experts and understanding of the rules (for example, Self, 1996). Although the generation of a conflict set of potential rules to fire makes the problem-solving process flexible, it can also be very inefficient and expensive computationally. To improve the efficiency, the Rete algorithm is used to construct a tree structure for the conditions in the rules which increases the speed of determining the conflict set. The restricted syntax of rule-based systems has disadvantages in that it makes it difficult to represent *structural* knowledge, i.e. knowledge about which entities are relevant in the domain. This is particularly the case where the domain is hierarchical in nature. Finally, in domains where the order of firing rules is important, updating and maintaining the system (adding or deleting rules) can be very difficult leading to unexpected results when the new program runs.

2.7.2 Case-based Reasoning

Case-based reasoning (CBR) is modelled on the way that humans use past experience to solve new problems (Kolodner, 1993). When confronted with a new problem, the user must identify the current situation and then consult memory to find a similar past case. If problems similar to the current one have previously been encountered, these are retrieved along with their solutions. The past cases are then ranked according to their similarity to the current case. We then choose a case and decide whether to reuse the solution or adapt it to fit the new situation. After evaluating the proposed solution, the system is updated to incorporate the latest experience. Alternatively, if no similar case has been encountered in the past, the new case must be accommodated in memory. Within our diagnostic system, a “problem” represents an incorrect answer to an algebra question and the appropriate “solution” comprises the identification of all underlying misconceptions. This paradigm thus provides a natural approach for constructing an intelligent diagnostic system for algebra by reconstructing the steps that students make when solving algebra problems.

Work on natural language processing at Yale by Schank and Abelson in the mid seventies led to the theory of *scripts* as a means of human knowledge representation. A script is a generalised experience recorded in memory (eg. we have certain expectations which are

evoked in memory when we hear a term like “party”). Related to the theory of scripts is the idea of *frames* as a means of representing knowledge. Frames were introduced by Marvin Minsky in 1975 and are similar to records in a traditional database with slots for entry of attribute values (initially these may be empty or may be set to default values). In both cases, the underlying theorem is that humans do not record full details of a situation in memory, but only need to record certain key differences.

The original work on CBR began in the 1980’s. Case-based reasoning involves remembering previous solutions of problems similar to the current one and either reusing the solution in the current case or adapting it to fit the differences between the old case and the current one. This is similar to the way that humans engage in problem solving: using past experience to guide them through the problem. CBR can be characterised in four stages: remind, retrieve, reuse and repair. The first stage is recalling past situations that are similar to the current one. This requires understanding of the current situation and an ability to analyse differences between past and present cases. If the current problem is *identical* to the past one, the previous solution could be reused. However, it is more likely that there will be some differences between the retrieved case and the current one and so some adaptation of the previous solution will be required. The new solution is then added to the case base.

The fundamental problem involved in CBR is how to *index* cases to enable efficient and appropriate retrieval. Storing failures (as well as successes) can be advantageous as failures tell us what *not* to do or what did not work in the past. The case-based reasoner “learns” by indexing new cases and adding them to its memory. Cases worth remembering are those that differ in some way from past cases. This requires that the system also store some normative knowledge and, to avoid redundancy, some way of determining whether the differences are significant enough to make the new case worth storing. Kolodner stated this as: *If the difference is instructive such that it teaches a lesson for the future that could not have been inferred easily from the cases already recorded, then record it as a case* (Kolodner, 1993). The implication for a diagnostic

system is to determine whether an error made by a student represents a misconception (in which case it should be stored) or a random slip (in which case it should not be stored), which is the subject of Repair Theory (van Lehn, 1983).

Of the two paradigms outlined here, case-based reasoning was chosen to implement the diagnostic system. A justification for this choice is now given.

2.7.3 Comparing Rule-Based and Case-Based Reasoning

The use of rule-based reasoning to implement diagnostic systems has both advantages and disadvantages. Firstly, the level of diagnosis and explanatory power provided by such systems is inadequate, because they cannot identify what a student was attempting to do when answering a question. They are capable of identifying broad areas of knowledge that a student has not mastered, but cannot identify specific misconceptions or patterns of behaviour. Secondly, systems are hard-coded; meaning that they can only deliver a finite number of questions and their knowledge is not transferable to other domains. The major advantage of this type of system is that there is a predetermined link between a particular question and the student's answer. Whilst this reduces search time, the resulting systems have limited explanatory power. Finally, search and retrieval methods are inefficient and adaptation is difficult with rule-based reasoning systems. It is anticipated that a system that is capable of identifying a student's solution strategy will be able to provide a finer granularity of diagnosis than that provided by existing systems.

Our choice of CBR as the paradigm for implementing the algebra system was based on several factors. Firstly, CBR enables us to use partial matching during the search and retrieval stages. This is important in our system because arithmetic slips could lead to an infinite variety of answers and hence prevent us from ever finding a perfect match for a particular answer on a given question. Because we are interested in diagnosing *algebraic* errors and misconceptions, we do not want to consider arithmetic slips. By considering the form of the answer, we can expect that arithmetic slips will pose fewer problems for matching in a CBR system than in a rule-based system.

Secondly, we want to identify the student's *cognitive* processes in solving algebra problems. These can be determined from the categorisation of a question and the choice of solution technique. This information will improve the explanatory power of the system and will also supplement the information recorded in the student's history, which is used to guide the depth-first search involved in matching answers on subsequent questions. Because an expert has well-distilled knowledge, their problem-solving behaviour tends to be both efficient and stable. Such behaviour can be well modelled using rule-based reasoning. However a novice is in the process of acquiring and accommodating new knowledge. As a result, their problem-solving performance cannot be expected to be stable. Trapping information about the outcomes for the different stages of problem solving supplements the user model and is useful to the teacher in helping them to tailor remediation for the individual, thereby assisting students in improving their mental models.

Thirdly, although individual "mal-rules" can be easily represented using a rule-based reasoner, it is much more difficult to look for the patterns of behaviour in such systems. As Sleeman noted, the incorporation of *families* of errors cannot be handled efficiently in rule-based systems (see Sleeman, 1984), but is much more straightforward within a case-based system. One of the most common error types that students make are Generalised Distribution errors where the student treats all operators and functions as though they were linear (see Matz, 1980 and 1982). This error type is encountered in all areas of mathematics including algebra, trigonometry and calculus. By including a template for Generalised Distribution in our solution case base, we can instantiate it in terms of a particular problem and thereby generate the corresponding error case.

Finally, but possibly most importantly, because of their structure, rule-based systems are notoriously difficult to adapt, whereas adaptation is fundamental to CBR. For example, when testing an early version of the rule-based Leeds Modelling System, Sleeman encountered a new mal-rule for solving linear equations (see section 2.3.1.2 for detail).

He expected that updating the system to accommodate this new situation should have simply involved adding one new rule to the system. Instead, he found that modifying the system “led to an explosion in the number of models to be considered, and so a reformulation was carried out”. One method for achieving cognitive diagnosis in a computerised system is to have a very limited set of questions and associated answers (correct and incorrect) hard-coded into the system. When a student inputs their answer it is compared with those available within the system. If there is an exact match between the student’s answer and one resident in the library, then we can be reasonably certain that we can reuse the information associated with the answer (eg. remediation). However, if there is no direct match, the knowledge engineer needs to intervene to incorporate the new answer into the system and, possibly, to change the entire structure of the system. As a result, automation and adaptation are not easily accommodated in rule-based systems.

This section has outlined two paradigms for implementing systems that are capable of automated reasoning, and explained the choice of paradigm used to implement the new diagnostic system for algebra. The next section provides an overview of the methods developed in the current research to realise the aim of improving cognitive diagnosis in the domain of tertiary-entry level algebra.

2.8 The New Approach to Cognitive Diagnosis

In this chapter, we have detailed problems that students encounter when making the transition form secondary-level to tertiary-level mathematics, reviewed a number of systems that have applied artificial intelligence (in particular, rule-based reasoning) to the modelling of performance in mathematics and explained the need for a new approach to cognitive diagnosis. The aim of the current research was to create such an approach and to evaluate its success by designing and implementing a diagnostic system for algebra. An outline of the new approach is now given.

An experienced mathematics teacher uses a combination of domain knowledge and knowledge of the processes that students employ during problem solving to deduce what a student was thinking when answering a question. The reason that they are able to do this is that student responses, even incorrect ones, are *not* random; rather, they are systematic and are related to the problem-solving techniques employed by students. Thus there is a "logical" path that can be followed to arrive at the given answer (Davis, 1984). The task for the teacher is to determine this path and then to remediate any misconceptions that the student holds. It is exactly this type of performance that is to be emulated by a cognitive diagnostic system.

To achieve this, the research has addressed several major issues. The first of these was to choose a model of human knowledge representation that could be replicated within a computerised system by adopting a suitable paradigm. The chosen model is that proposed by Davis (ie. the use of schemata) and case-based reasoning has been selected as the most suitable implementation paradigm. The reasons behind these choices were outlined in sections 2-6 and 2-7 above.

The next step in the research was to develop a process for diagnosing algebra errors that is based upon the identification of the solution strategy that was most probably employed by the student. This required the development of a taxonomy of errors that identified and exploited the relationship between a student's solution technique and the **structure** of their resultant answer. The need for this taxonomy arose because, whilst the literature contains details of the types of errors that students make when solving algebra questions, the errors had not previously been categorised in a manner that enabled a computerised system to automatically determine the link between an answer and the solution path that gave rise to it. To develop the taxonomy, detailed analysis of erroneous answers was conducted (see chapter four).

Development of the taxonomy of errors led to an explicit relationship between solution techniques and the **structure** of the resultant answer. However, to exploit this

relationship in a computerised system, methods for linking questions, erroneous answers and an **explanation** of the underlying misconceptions had to be developed. This problem was resolved by developing a model of algebraic problem solving that employs the same steps that humans undertake during general problem solving (Polya, 1948).

Evaluation of the new methodology was conducted by designing and implementing a diagnostic system that is based upon this new methodology for cognitive diagnosis. The diagnostic system is capable of setting algebra questions in standard form (expansion and factorisation), using a computer algebra system to generate the correct answer, receiving the student's one-line answer, marking this and, if it is incorrect, it is then diagnosed. The system is fully automated, so the teacher is not restricted to using a predefined data bank of questions. This may not be so important here, but if the methodology is to help in improving student modelling (by better diagnosis) then it must not be restricted. The system uses identification of the solution strategy as the key to effective diagnosis. It is capable of **partial** matching on answers, which is important if we are to avoid being entangled in the infinite array of arithmetic slips that can be made during the execution of a particular strategy. Thus, the *form* of the student's answer provides the best indication of which problem-solving method the student adopted and hence reveals information about the student's internal knowledge structures. By tracing the student's path through the question, the system can provide both an explanation and remediation for the errors.

The system has two main phases of operation: the classification of questions and the subsequent diagnosis of erroneous answers. However, other processes also needed to be developed to enable the system to proceed automatically. These included the development of a parser that can decompose algebraic expressions into their structural components and the development of a new method for marking answers. This latter requirement was born of the need for the system to be able to recognise expressions that are mathematically equivalent. Typically, other systems use one of two methods for marking answers - either the evaluation of expressions over a grid of points or the use of a computer algebra system. However, both of these approaches would regard the expressions $(x+y)(x-y)$ and

x^2-y^2 as being identical, but for the purposes of identifying different solution techniques, we required a method that can distinguish between the two expressions (see chapter six). Evaluation of the system's performance showed that it is very accurate when classifying questions and when utilising answer **structure** to diagnose errors.

2.9 Summary

This chapter has provided the background to current research, which is aimed at improving cognitive diagnosis by developing a methodology that is finer-grained in its diagnosis and has a greater ability to explain its reasoning processes than existing systems. The need for this was provided by the evidence of insufficient mathematical understanding exhibited by students entering tertiary studies and by the methods that universities have adopted to address the situation. Several approaches to the application of artificial intelligence in the domain of mathematics education were also reviewed. This showed that existing techniques for modelling student behaviour are successful for procedural problems, ie. those that can only be solved by one solution technique, which is explicit in the problem (eg. multi-digit subtraction or algebra problems such as *Expand* - 2(4*x* - 3*y*)). This success is due to the fact that the skills required to solve such problems are immediately identifiable, however for problems that can be solved by a variety of techniques, the skills required for solving are determined by the chosen solution technique. Thus, this technique must be determined before further diagnosis can proceed.

The research required relevant research findings from the areas of mathematics education and artificial intelligence. Classroom studies of students learning algebra helped in the development of models of mathematical problem solving, the error-making process and of human knowledge representation. The manner in which humans encode their knowledge is subject to great debate between cognitive psychologists. Two competing theories are schemata (similar to frames, with slots for variables) and productions (rules of the form IF {condition} THEN {action}). There is evidence to support both of these

models, which probably implies that we use a combination of these methods, depending upon the knowledge to be encoded. For mathematical knowledge, this author contends that the major component of knowledge representation is a **schema**, which includes information in a variety of formats (pictorial, verbal etc.). This is in line with the method of mathematical knowledge representation proposed by Davis (Davis, 1984) and of the processes of problem solving, which have been used to develop a model of algebraic problem solving. Chapter three contains details of the four-stage model and relates it to the model of knowledge representation adopted here.

As students learn, they expand their existing schemata and construct new ones. This activity is error-prone, but Matz's analysis of errors and the techniques that produce them provides evidence that supports schemata theory and that accounts for the types of algebra errors commonly made by students. However, a taxonomy of errors had to be developed to support an automated diagnostic system. Chapter 4 of this thesis provides details of the error analysis undertaken in the current research, the resultant error taxonomy and evidence that supports schemata theory. Further, it shows that the chosen implementation paradigm (viz. case-based reasoning) can achieve greater explanatory power than rule-based systems can.

The remainder of this thesis is divided into six chapters. Chapters 3 and 4 present the model of algebraic problem solving and the taxonomy of algebra errors that were developed as a means of realising improvements in cognitive diagnosis in the domain of algebra. To evaluate the diagnostic capability of these methods, a diagnostic system for algebra was produced. Chapter 5 contains details of the design of the system and chapter 6 describes how the system has been implemented. Details of the evaluation of the system's performance and possible improvements are provided in chapter 7. In the final chapter, we review the success of the new approach to cognitive diagnosis as exemplified by the implemented system and detail the contribution made by this research.

In summary, the important features of this research are:

1. Choice of a suitable knowledge representation scheme (schemata); this is related to the choice of a suitable implementation paradigm (case-based reasoning),
2. Development of a model of algebraic problem solving that is used to guide the performance of the diagnostic system and hence to increase its explanatory power,
3. Development of a taxonomy of errors to enable automated diagnosis, and
4. Development of processes for decomposing algebraic expressions and for marking answers.

Chapter 3 A Model of Algebraic Problem Solving

Cognitive diagnosis is a complex task that is concerned with describing the mental processes employed by an individual learner on a particular task. Complexity arises because learners range in their levels of expertise and the individual learner's performance may not be a stable process (Sleeman, 1984, Siegler, 1987a, 1987b, Payne and Squibb, 1990). Improving competence in mathematical problem solving requires the learner to construct both comprehensive knowledge structures and reasoning processes. As a student constructs more knowledge and begins the transition from novice to expert, their problem solving behaviour varies, but this is not uni-directional - students can forget previously acquired knowledge and their problem-solving behaviour is influenced by contextual information and can regress under cognitive load (Sweller, 1992). This means that simply because a student has shown mastery of a particular concept in one question it cannot be assumed that they will subsequently always demonstrate the same ability.

Problem solving is commonly used to measure the effectiveness of teaching and learning because it reveals information about the learners' concepts of the task and their ability to transfer their knowledge to the solution of novel problems. Mayer identified five types of knowledge that humans use when solving mathematical problems, including linguistic knowledge (knowledge of language), semantic knowledge (knowledge of facts of the world), schematic knowledge (knowledge of problem types and applicable solution techniques), procedural knowledge (knowledge of how to perform the steps in a solution technique) and strategic knowledge (knowledge about the problem-solving process) (Mayer, 1983).

Errors in problem solving therefore have several potential sources and, ideally, a cognitive diagnostic system should attempt to identify as many of these sources as possible. However, existing diagnostic systems typically only identify errors arising from one such source, viz. failure on the behalf of the student to correctly execute steps in a solution technique. That is, they identify errors in the student's "know-how" but are incapable of identifying "know-when" errors, which leads to coarse-grained diagnosis. However it is not enough to patch errors made when executing a particular solution technique, students must also be guided to choose the most efficient and appropriate techniques. Instead, determining what strategies students have employed can lead the teacher to a better understanding of why learning takes the form that it does (Siegler, 1987a). A second major problem with existing diagnostic systems is their limited power to explain the reasoning that led to a particular diagnosis. One reason for this is that, for problems that can be solved using a number of different approaches, the systems are only capable of determining whether the correct answer was given and they cannot identify the particular solution technique that was applied.

To improve the explanatory power of a cognitive diagnostic system, its design should be based upon a model of problem-solving behaviour. This chapter introduces the model of algebraic problem solving that has been developed for this purpose. Polya defined a general model of human problem solving comprising four stages: *interpreting* the problem, *planning* a solution, *executing* the plan and *reviewing* the success of the plan (Polya, 1948). This methodology provides a useful description of the process of problem solving, but it is too vague to be directly implementable and fails to take account of variations in the level of expertise of different problem solvers. The proposed model of algebraic problem solving was derived from extensive analysis of the behaviour of many students in algebra tests (see chapter four for full discussion), and represents a computationally feasible extension and modification of Polya's model. It allows us to determine at least some of the conceptual activities undertaken by students when solving algebra problems, not only the procedural ones. Further, it has enabled us to implement the four steps in a manner that can be adapted to the level of the individual student and that reflects the recursive nature of problem solving.

In this chapter, the processes involved in algebraic problem solving are outlined and the effects of factors such as the individual's knowledge representation and expertise are demonstrated. This work is then discussed in terms of the computational model of algebraic problem solving that was developed as the basis for the design of the algebra diagnostic system (the full details of which are provided in chapter five). The reasons for adopting this approach to modelling student problem solving are also discussed.

3.1 A New Approach to Modelling Algebraic Problem Solving

The purpose of all good teaching (whether this is delivered by a human or a computerised tutor) is accurate and efficient learning. This is usually measured, both in the classroom and computerised systems, by testing. However, it is not enough for a student to answer questions correctly; they should also be aware that many problems have multiple solution techniques available and they should choose efficient and appropriate techniques. Ideally, students should construct new knowledge efficiently and accurately, and this should occur without "slippage", that is, once a student has mastered a given fact or procedure, they should subsequently apply the new knowledge correctly and this should not lead to a degradation of the pre-existing knowledge. However, learning is rarely uni-directional; instead when a student is presented with a novel problem, they can revert to earlier problem-solving behaviours, particularly as the cognitive load for a given task increases (Sweller, 1992).

Classroom studies have led to a large degree of agreement between cognitive scientists with regard to the manner in which students solve mathematical problems. Solving routine mathematical problems requires the student to select a schema, instantiate it and then execute the solution strategy. This is in line with Polya's four-stage model of general problem solving: *interpret* the problem, *plan* a solution, *execute* the plan and *review* the success of the plan (see Figure 3.1). For an expert, the main step in the problem solving process is recognition of the form (or category) of

the problem (ie. pattern matching), which leads to the retrieval and implementation of a solution strategy. The problem environment provides cues or stimuli for the recognition and retrieval stages. Recognition of the form of the problem consists of determining the problem category whilst retrieval returns a solution strategy based on the question category. The main difference between novices and experts is that experts have such strong recognition of problem types that they can combine the categorisation, planning and retrieval phases whereas novices must engage in a search for a solution technique (Cooper and Sweller, 1987, Chi *et al*, 1988, van Lehn, 1989, Snyder, 2000).

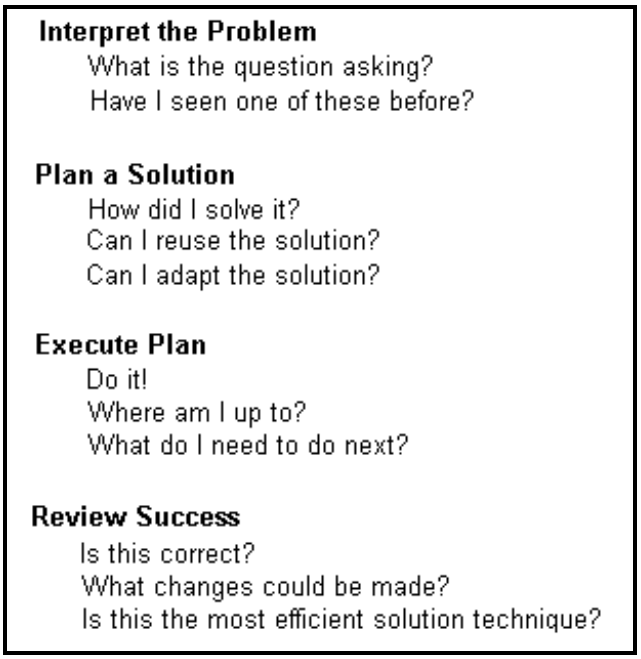

Figure 3-1 Polya's Model of General Problem Solving

To interpret a problem and to plan a solution strategy, students use a form of *analogical reasoning*, by matching the current problem situation with a familiar one stored in memory (Genesereth, 1982, Davis, 1984, Anderson, 1990, Low and Over, 1992, and English, 1997). To access their stored knowledge, students must form a mapping between the current problem and existing representations (or schemata) in long term memory. Once the mapping has been created, the stored problem can then be recalled along with its solution, which is adapted to fit the new problem.

During algebraic problem solving, the mapping is achieved in two stages that are based upon the goal of the problem, the problem environment and different problem states achieved during the solution process. The first of the two stages is a parsing or *categorisation* problem. The inputs to this stage are the goal (eg. *Solve*) plus a set of features extracted from the problem expression. The choice of which features to extract from a problem depends upon the individual student's interpretation of the problem, their beliefs about the importance of features and their familiarity with operations that can be applied to achieve the necessary steps (Davis, 1984). The second stage is a *selection* problem, where the student combines the result of the categorisation stage with the (same or extended) set of features to determine the most suitable solution plan. Errors in algebraic problem solving occur when the learner either applies an inappropriate solution technique or incorrectly adapts an existing one to fit the current problem (Matz, 1980).

Capturing the student's cognitive processes is the most important aspect of diagnosis. Cognitive processes include assessment of similarity between the current problem and ones previously solved by the student and the choice of which solution plan to apply to a particular problem. They therefore also determine the errors made when executing the chosen solution plan. As a student constructs greater knowledge structures and begins to make this transition, their problem-solving behaviour varies because it is influenced by contextual information and can regress under cognitive load (Davis, 1984, Sweller, 1992). A successful diagnostic system should be able to identify the student's solution technique, which means that it must be able to mimic a student's perception of similarity. This perception is dynamic and is determined by the student's internal knowledge representation (van Lehn, 1989). Depending upon which features of an algebra problem students perceive to be relevant, they may categorise a particular question in a number of different ways. This then impinges upon their choice of solution technique, and in turn determines which errors they may make.

Because novices have a limited number of schemata, the problem features that assume greatest importance tend to be *superficial* and limited in number (Gentner, 1989,

Ross, 1989, Pierce and Gholson, 1994). There are several possible reasons for the choice of features. It is possible that the chosen features are determined by the student's ability to read text and those textual features that are most conspicuous (Julesz and Bergen, 1983). For example, in algebraic problems, indices and brackets are highly salient (due to their shape, size and location) and hence assume greatest importance. It is also possible that students recognise that these features have highest precedence in problem solving and hence provide a natural starting point for solving a problem. Regardless of the reason, it is true that novices concentrate on these features and tend to leap straight to a solution technique, which will often rely on brute force and will be far from the optimal technique available (see chapter four for more detail). On the other hand, experts have more schemata and spend more time analysing the *relational* features of the problem to categorise it (ie. determine its question type) and then to choose a solution technique which is (close to) optimal (Chi *et al*, 1988, van Lehn, 1989). As students learn, they construct more schemata and spend more time in analysing problems. However, their methodology can regress under cognitive load and they can then revert to earlier, more primitive processes.

The key to effective diagnosis is to determine how the individual has represented a particular question and to use this to guide the searching and matching phases. The implication for a computerised diagnostic system is to be able to replicate the methodologies of both novice and expert problem solvers, and to determine which methodology a particular user has adopted. An individual student can operate at different levels of expertise at different stages within a single algebra problem. Such a shift in the operational expertise level is usually associated with a student who encounters a change of goal during the solution process (see chapter four). The system developed as part of the current research is capable of modelling inconsistency in behaviour by reconstructing a solution path for a given question and flagging the expertise level for each step in the solution process.

Before providing details of the computational model, knowledge representation and the nature of problem-solving expertise and its impacts on algebraic problem solving are discussed.

3.2 Cognitive Models, Knowledge Representation and Similarity

In this section, we describe the model of knowledge representation upon which the diagnostic system is founded and outline how this impacts on a student's perception of similarity between algebra problems.

3.2.1 Representing Mathematical Knowledge

As students learn, they construct new knowledge which must be incorporated into their internal knowledge bases. Ideally, the student's internal structures should mirror the overall domain structure, which is strictly hierarchical for algebra. The model of knowledge representation upon which the current work is based has been derived from a number of cognitive models. The basic unit of knowledge is a *schema* (Sweller and Cooper, 1985, Cooper and Sweller, 1987, Davis, 1984 and Pirie and Kieren, 1994, Chinnappan, 1998), which is a cognitive construct that enables us to perform pattern recognition and to categorise problems according to the moves required to solve them. A schema is like a template that contains slots to be filled either by input values or default values (Schank and Kolodner, 1979). The knowledge encoded here includes verbal, symbolic and graphical information as well as the procedures available for solving a particular problem (see Figure 2-10). Novices have few of these whereas an expert may have thousands.

Along with schemata, Davis also proposed that humans have *critics* stored in memory. A critic is a different form of knowledge representation structure which enables us to create analogies and to determine how reasonable an answer is. Critics are used to guide the selection processes for both classification and planning, as well as for monitoring progress during the execution of the chosen solution strategy. Again, experts have many critics, whereas novices have very few. The absence of critics can lead the novice either into incorrectly adapting a known rule to fit the new problem or

into failing to understand the significance of particular numbers (eg. the zero on the right-hand side of the equation x^2-5x - 6 = 0, see Matz, 1982).

Algebraic problem solving requires the student to identify what the question is asking (classification task), planning a solution and then executing the plan. The student's schemata will provide the information required for the first two tasks. The categorisation task may result in a number of interpretations. Experts perform classification on the basis of the moves needed to solve a problem, whereas novices are more affected by surface similarities (see section 3.3 for more detail). Novices, on the other hand, have only a limited number of schemata, and these will lack both the detail and connectivity of an expert's schemata. Generalisation and abstraction, which are required to improve connectivity between schemata, only occur with considerable practice and exposure to a wider range of schemata. As a student develops more understanding, they develop "super-operators" which combine several problem-solving steps into one (called "compounding"), and they also adapt their selection heuristics (called "tuning"). Newell and Rosenbloom used the term "chunking" to refer to these two processes (see Davis, 1984). We now demonstrate the impact that a student's knowledge structures can have on their problem-solving behaviour.

3.2.2 The Impact of Knowledge Representation on Perceptions of Similarity

Factorisation problems can be solved by adopting the following rule-based approach, which is typically included in secondary-level textbooks (for example, Nolan *et al*, 1994). The steps are:

- remove any common factor,
- if there are two terms, look for specialised cases such as difference of two squares, difference of two cubes or sum of two cubes (uses term types and signs on terms),
- if there are three terms, look for quadratic trinomials (uses term types) or grouping problems,
- if there are four terms, look for cubics (uses term types) or grouping problems, and
- check that the problem is fully factorised.

This method requires the problem solver to recognise and remove a common factor, if one exists. If a common factor is removed, focus is shifted to the bracketed term, otherwise focus remains on the complete expression. With reference to the (reduced) problem, the number of terms in the expression to be factorised becomes the most important (or discriminating) feature, followed by the term types and then by the signs on the terms. This approach can be represented by the discrimination network in Figure 3.2, which contains redundant nodes (for example, there are six nodes for *General Quadratics*). Full details of modelling algebra questions using discrimination networks are contained in the appendices.

Whilst this approach to factorisation is successful, it is not efficient and is unlikely to be employed (see chapter four for full details). So, how do students approach categorisation and what are the features they use? Powers are highly salient due to their size and location which makes them visibly conspicuous; this is reinforced by the fact that exponentiation has the highest level of precedence. For these reasons, quadratic and cubic expressions are readily noticed by novices and experts alike. Thus it seems more reasonable that classification will be performed using *term types* as the most important feature and a second classification tree for factorisation problems is therefore shown in Figure 3.3. Because this tree groups related problems into families (such as *General Quadratics*), it embodies the hierarchical nature of the algebraic domain and hence has no redundancy. It also reflects the differences in the solution strategies employed by students operating at different levels of expertise.

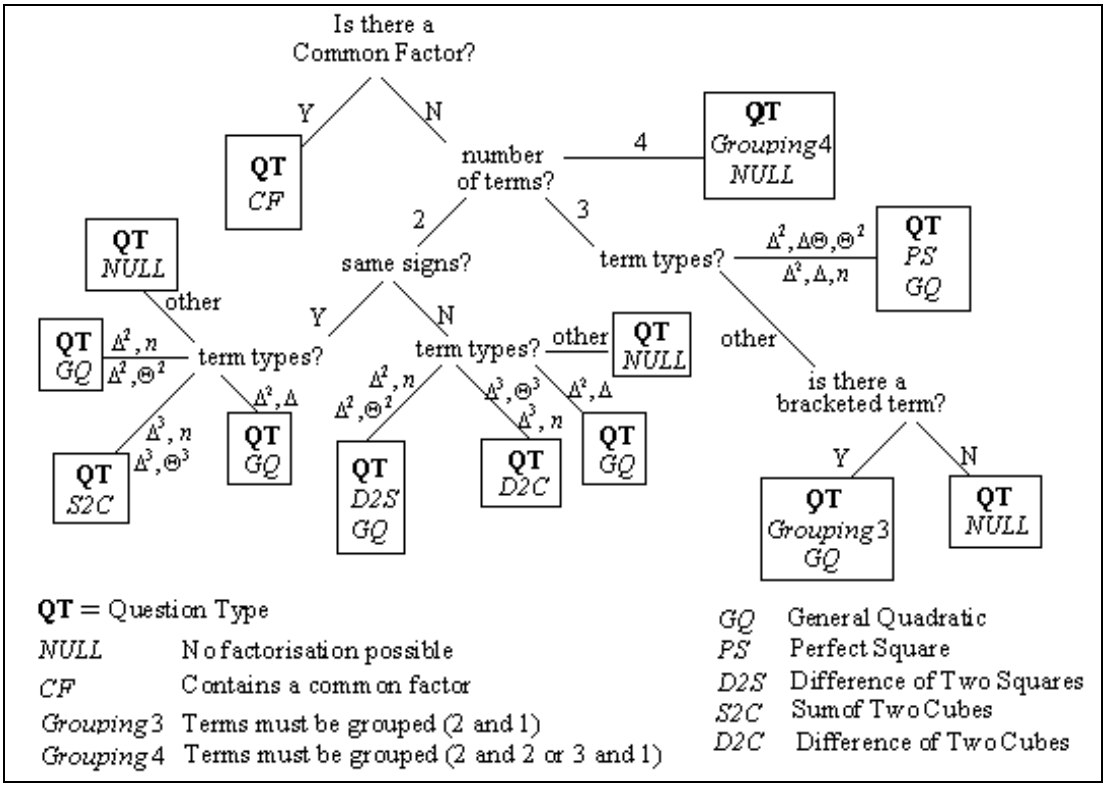

Figure 3-2 Rule-based approach to classifying factorisation problems

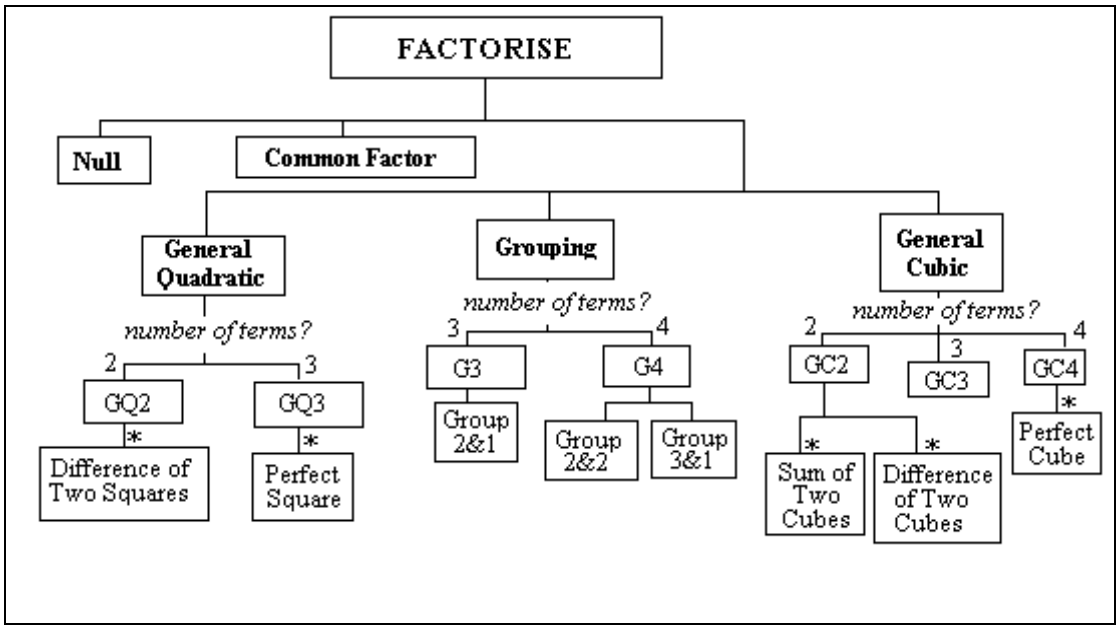

Figure 3-3 Classification Tree for Factorisation Problems

Experts are aware that within families of problems, there exist very specific cases with their own specialised solution strategies (such as the template for a Difference of Two Squares problem). That is, they have deep, well-developed classification trees, which

they traverse in two stages. The first stage is a breadth-first search, which uses the term types to identify the *family* to which the current problem belongs. The second stage is a depth-first search (within the family), which uses other features (such as the number of terms and the signs of the terms) to identify the narrowest class to which the current problem belongs.

Novices, on the other hand, have very few schemata of algebraic problems and hence their classification trees have very few levels resulting in classifications which are general rather than specific. Thus a novice is likely to identify the broad family to which a given problem belongs, but is unlikely to make finer distinction. This results in them adopting solution strategies that can be very inefficient and labour-intensive (see section 3.3 for detail). This is not to say that a student operating at a novice level for most problems will never classify at a finer level of granularity. For example when asked to factorise the expression a^2 - b^2 , most students in year nine or beyond will recognise this as a difference of two squares problem and apply the associated template to obtain the correct answer $(a + b)(a - b)$. However, when asked to factorise the expression $(3a+5b)^2$ - $4b^2$, very few students apply the template, indicating that they have not categorised the problem as finely as possible (see Figure 3.4). The solution techniques that these students apply indicate that if they have explicitly categorised the problem, they have done so correctly but very generally (ie. as a *General Quadratic* problem). In many instances, students appear to believe that this problem is completely novel (ie. does not fit any of the general categories) and attempt to solve it by using a means-end approach. That is, they expand the bracketed term in the hope of transforming the overall problem into a more recognisable form.

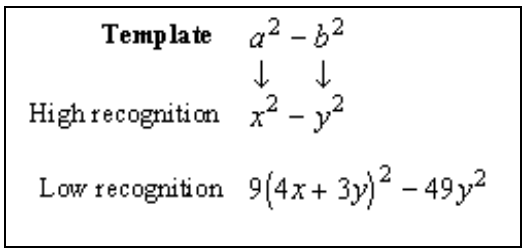

Figure 3-4 Template Recognition

Assigning values to the variable *Term Types* is not enough for full classification of a factorisation problem. A student's recognition of patterns is affected by the detail of a problem (greater detail results in higher cognitive load and broader classification) and the manner in which they *expect* to see a problem. The vast majority of textbooks and classroom teachers present difference of two squares problems using pronumerals and with both coefficients equal to one (ie. in the form $a^2 - b^2$), without making it explicit to students that the associated template applies to any factorisation problem that contains two terms that are both perfect squares but which have opposite signs. The emphasis in this approach is not on the *structure* of the problem and hence transference is hindered. Thus students can readily identify difference of two squares problems presented in standard form, but recognition quickly degrades when the problem contains greater detail.

The next section discusses the nature of problem-solving expertise and explores the relationship between expertise and knowledge representation. The discussion of expertise is limited to those characteristics which might be expected to be observed in the domain of solving highly structured algebra problems.

3.3 The Nature of Problem Solving Expertise

The aim of good teaching is effective learning; that is, we not only want students to correctly solve problems, but we also want them to adopt appropriate and efficient solution strategies. The effectiveness of teaching and learning are commonly measured by testing performance in problem solving, however the emphasis is usually placed on whether or not a student can generate the correct answer and does not take account of all the processes involved in problem solving. Better insight into a student's performance is gained when we also consider how students interpret problems and their choices of solution techniques.

Problem-solving performance changes as the subject spends more time engaged in problem solving and constructs new knowledge. Hence examining problem-solving

performance provides insight into how students learn. For these reasons, numerous studies in cognitive science have focused on expertise in problem solving in domains as diverse as game-playing (for example, chess and Go), classification tasks (for example, card sorting) and solving scientific problems (in domains such as physics and mathematics). The results of these studies have enabled cognitive scientists to identify a number of features that distinguish experts from novices (for example, see Chi *et al*, 1988, Van Lehn, 1989 and Snyder, 2000).

Within the domain of solving structured algebra problems, those characteristics that most strongly differentiate between expert and novices are:

- experts are faster at solving problems than are novices,
- experts are more accurate than novices,
- experts are stable in their problem-solving behaviours whereas novices are more affected by contextual information and can be erratic in performance,
- experts adopt a top-down approach to problem solving whereas novices focus on small parts of a problem,
- experts tend to focus on relational features of a problem whereas novices focus on superficial features,
- experts are better than novices at estimating the degree of difficulty of problems than are novices,
- experts have more strategies available and are better at assessing the value of these than are novices,
- experts have greater episodic memory (ie. are better at recalling the details of specific episodes from the past),
- experts are good at monitoring their own progress and correcting their own errors,
- experts are more likely to review the results than are novices,
- experts are better at recognising meaningful patterns than are novices,
- experts have better recall (short-term memory),
- experts *see* more than do novices (ie. their knowledge is greater and "chunking" is better).

If we map these characteristics onto the domain of algebraic problem solving, then we would expect to make the following observations:

1. an expert mathematician should be better at recognising problems that are functionally related than a novice is,
2. an expert mathematician should be aware that multi-step problems can be solved by a variety of techniques, but they consistently choose the most efficient techniques, and
3. an expert mathematician should be both faster and more accurate than a novice when solving algebraic problems.

All of these expectations have been observed (see chapter four) and we now examine the relationship between a student's level of expertise and their performance at the different stages of algebraic problem solving. (Note that the term *expert* is usually reserved for a subject with *thousands* of hours' experience in the domain. Thus we cannot expect that any learner could be regarded as an expert, although at times they may demonstrate problem-solving behaviours that could be dubbed "expert". Subsequently, when we use the term expertise to describe a student's problem-solving behaviour, we mean their "operational" expertise, which may vary from one problem to another. That is, the expertise level pertains to the solution technique, not to the student.)

3.3.1 Interpretation/Classification

Interpreting an algebraic question requires the subject to apply their knowledge of the domain to classify the problem. A form of analogical reasoning is used to assess the similarity of problems. An analogy is a mechanism for creating the mapping between two related problems. This calls upon the subject's ability to recognise meaningful patterns, which is in turn affected by their knowledge base (both in terms of its content and structure). If we make the assumption that the student is rational, then we can conclude that if a student has classified a problem very finely, then they will adopt the most appropriate solution technique available for that class of problem. For example, if a student has classified a problem as being a *Difference of Two Squares*

problem, then we can assume that they will apply the solution template rather than explore less efficient techniques. Hence we can deduce how the subject classified a problem by identifying their choice of solution technique.

Schema selection requires that the student have some means of measuring the similarity states as well as some heuristics for choosing between contenders. The classification of problems by novices is based on literal, surface features whereas experts employ relational features and chunking, which enables the perceptual system to rapidly parse stimuli to generate a hierarchy of instantiated chunks.

We now examine two algebra examples to demonstrate the differences in the behaviours of experts and novices on the classification task.

Example 1 Factorise $(3a+5b)^2 - 4b^2$

This example was considered earlier (see section 3.2.2), where we discussed the output of the classification task in terms of the student's ability to recognise patterns. One reason for the differences in pattern recognition is the structure of the subject's knowledge base. An expert has deep, hierarchical classification trees, whereas novices tend to have flat structures (see Figure 3.5). To improve problem solving performance, students need to acquire greater knowledge and to be able to access this quickly and accurately. This in turn drives the need to develop better information structures.

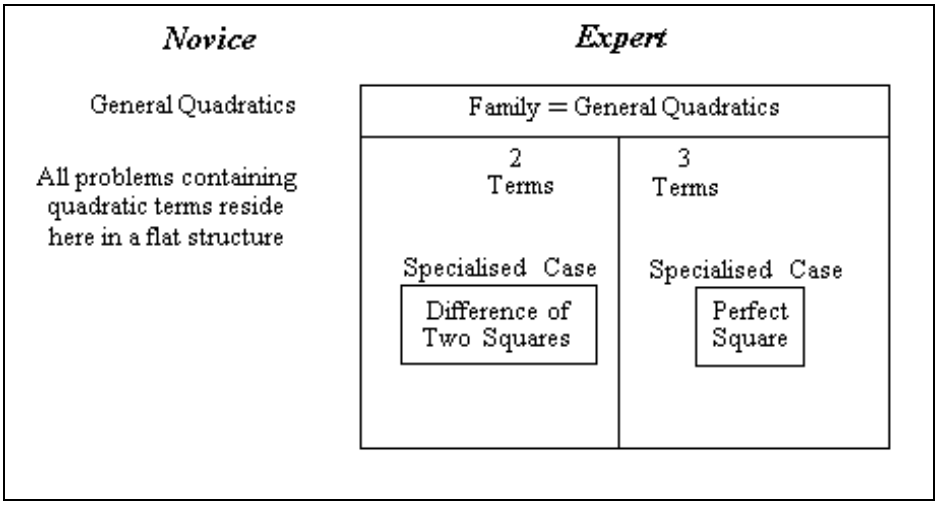

Figure 3-5 Hierarchical Structure of Knowledge

At this point, we draw an analogy with the operation of an office secretary. If the secretary only deals with a very few customers and files, there is no need to impose structure on the filing system - all files can be thrown into a single drawer. When a particular file is required, the secretary performs a linear search on all the files to locate the required one. However, as the number of customers and the number of services provided by the office increase, this mode of functioning becomes very inefficient. At this point, some structure needs to be imposed on the information. Files will be grouped according to the services (for example, payroll, customer services etc.) and then further subdivided by customers. This is a needs-driven operation.

Similarly, it is only when a student becomes aware of the proliferation of algebraic techniques and their interrelationships that they begin to restructure and reorganise their knowledge. In mathematics, this requires the student to make generalisations about problems and the associated solution techniques. They can then begin to create abstractions (in terms of the relational features of problems) and organise their knowledge more efficiently. Thus, even though an expert mathematician has a much vaster knowledge base than does a novice, they can locate and retrieve relevant information much more quickly.

Another characteristic of problem-solving behaviour that varies with expertise is the amount of time that the subject spends in analysing a problem. Because experts have well-organised knowledge structures, they can use features of a problem to serve as stimuli during the classification stage. They will use both surface and relational features to achieve this. The surface features are the same as those used by the novice (for example, presence of powers) but are only used at the first stage of classification; that is, these features trigger remindings of the general family to which the given problem belongs. A deeper analysis of the problem, resulting in a finer classification, depends upon the relational features of the problem (such as the number of terms, term types and signs on terms). Thus classification of problems is (at least) a two-stage process for an expert, but a novice will make (at most) one pass through the classification stage (see Figure 3.6).

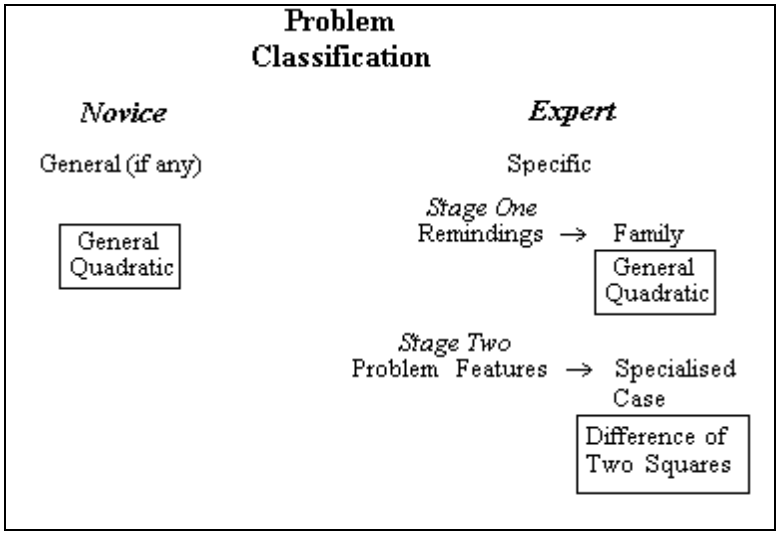

Figure 3-6 Expertise and its impact on classification

Example 2 *Solve* for t: $e^{2t} - 5e^t + 6 = 0$

When an expert approaches this problem, they will transform it into the equivalent problem *Solve* for x: $x^2 - 5x + 6 = 0$ where $x = e^t$. This is recognised as a general quadratic equation (in the variable x) which can be solved by first factorising the expression on the left-hand side and the applying the Null Factor Law. To solve for the variable t, the subject then applies their knowledge of related (inverse) operations. In fact, because experts use compounding to create their own "super-operations", they

may never explicitly make this transformation (see Davis, 1984). However, a novice lacks the knowledge required to identify the *need* to transform the problem. Instead, novices focus on the power terms and classify the problem as a General Exponential equation, which requires use of logarithms for solving (see chapter four).

In summary, we make the following observations about the performance of experts and novices on the task of classifying algebra problems.

- Experts classify problems more finely than do novices.
- Experts classify problems on the basis of the moves required to solve them.
- Experts have more schemata than do novices, and an expert's schemata are more detailed than are those of novices.
- Experts' knowledge bases are better organised than are those of novices.
- To improve both problem-solving performance and their knowledge structures, students need practice in the classification task. Emphasis needs to shift from the *execution* of solution procedures to the *selection* of the most appropriate techniques.

We have already demonstrated the relationship between the classification and planning tasks involved in algebraic problem solving (see section 3.2). This is now described in more detail.

3.3.2 Solution Planning

Experts spend much of their time in analysing a problem where they focus on larger concepts rather than on small, independent items (Sweller and Cooper, 1985). They also have large knowledge bases, are skilled at abstraction and generalisation, have well-developed pattern recognition and they use this to reduce memory search. Because of these superior mental structures, experts form an overall view of the problem and can categorise problems at a very fine level of granularity. This in turn means that they are aware of a number of contending solution strategies, and choose the most efficient and relevant technique. That is, an expert forms an overall solution strategy at the outset and usually executes this accurately. The mechanism fails if the

student lacks the appropriate schema, has an incorrect or incomplete schema, or retrieves the wrong schema.

On the other hand, novices are not as perceptive as experts and so do not consider the problem as a whole, but can only focus on parts of the problem. Novices have so few schemata available that they may not categorise the problem at all, but are very keen to do something. (This may be a by-product of classroom teaching where the emphasis is traditionally placed on executing solution techniques for blocks of related problems (although this is rarely spelt out), and little time is spent on practising categorisation and identifying what it is that makes some problems related but not others.) Even though they may have no starting point for the complete problem, students will often work on a part of it and review their progress. For these reasons, they tend to adopt a means-end approach when planning a solution strategy. This can be described as a three-step process:

1. Identify initial problem state,
2. Identify operators that can be used to transform the state of the problem,
3. Test to see if the current state is equal to the goal state. If so, stop, otherwise return to first step.

To apply this method, the student must be able to calculate the difference between two problem states. They must also determine those operators that will reduce differences between two problem states and apply heuristics to choose the best operator. In turn, they apply the operator and review the state of the problem. This method is labour-intensive and requires a great deal of over-writing of short-term memory, which can lead to errors. Figure 3.7 shows the effect that expertise has on solution planning for the factorisation problem from section 3.2.2.

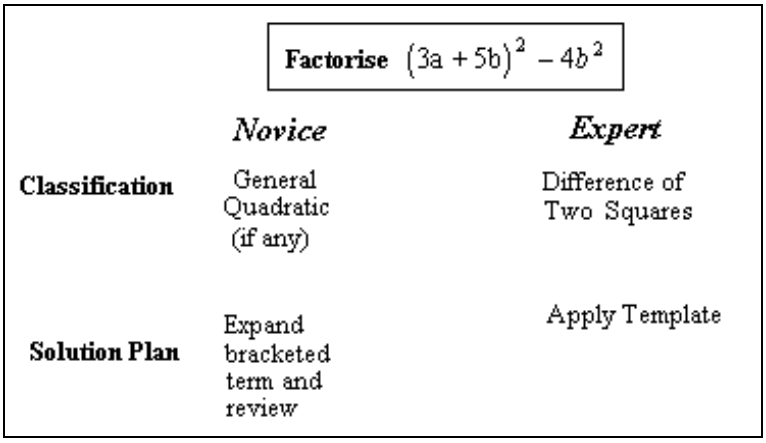

Figure 3-7 The effect of expertise on solution planning

In this particular example, the novice approach, whilst very labour-intensive, can lead the student to the correct answer. However, if the expression to be factorised is changed to $(3a+5b)^2 - 4c^2$, the means-ends approach will actually result in a problem state that is further from the goal state than the initial state was. This example emphasises the need for students to be encouraged to avoid the means-end approach to problem solving. The fact that novices focus on small parts of a problem combined with the means-end approach can result in meaningless answers. For example, consider the following factorisation:

$$x^2 + 6xy + 8y^2$$
$$= x(x + 6y) + 8(y^2) \quad .$$

Here the student has recognised the need to group some terms and to include brackets, but cannot work with the overall expression. However the trigger is provided by the keyword and so the student attempts to use brackets by grouping the first two terms. Once the student has produced some bracketed terms, (s)he feels satisfied that the answer is complete and correct.

One implication of the research into schema acquisition is that the development of expertise in a knowledge-intensive area such as mathematics requires the acquisition of an enormous number of schemata (Sweller and Cooper, 1985). This means that a

mathematics expert may be able to recognise tens of thousands of problem states and it is this knowledge that distinguishes an expert from a novice. Learning the rules of mathematics may not be difficult but learning when and how they should be applied requires the acquisition of many schemata, which may take many years.

Consider the problem: $\frac{a+b}{c}=d$, solve for a. An expert would immediately recognise that to solve this problem they should begin by multiplying both sides of the equation by the denominator; this move is generated by the relevant schema. In contrast, novices who are familiar with the appropriate rules of algebra will need to use a means-end strategy whereby differences between each given state and the goal state are extracted and the rules of algebra are used to attempt to eliminate the differences. In other words, novices need to search for a solution while experts can use their schemata to generate the solution without engaging in the search process. As knowledge increases, so do the number of schemata, the detail of the schemata and the connectivity between them. The presence of these may "substantially explain the differences between novices and experts" (Sweller and Cooper, 1985).

The selection heuristics used by experts and novices are different. Apart from their knowledge of the existence of many solution techniques, experts are also very good at judging the appropriateness of contending solution techniques. Their selection is guided by their critics (heuristics which are based on the structure of the problem), whereas a novice is more likely to be influenced by factors such as ease of use, frequency of use, recency of use and past success (for example, see Anderson, 1990). Errors made at this stage of problem solving are called *conceptual* errors, because they are related to the cognitive activities undertaken by the subject.

Matz explained conceptual errors as the results of a student attempting to adapt previously constructed knowledge to a new situation (see Matz, 1980, 1982, and Chapter 2). Two of the most common extrapolation errors that students make are generalised distribution errors and repeated application errors (these errors were discussed in detail in section 2.5).

In summary, experts adopt a top-down approach to analysing problems. Because they have better perception, recall and episodic memory than do novices, they are better at recognising when to apply templates. They are stable in their problem-solving performance and consistently use the most appropriate solution techniques. We now review performance in executing solution plans and how this is affected by expertise.

3.3.3 Execution of Solution Plan

At the start of this section, we noted that experts are both faster and more accurate at solving problems than are novices. In this section, we examine the reasons for this observation in terms of the chosen solution strategy and the subject's self-monitoring skills. We also consider two types of executive errors, viz. *procedural* errors and *control* errors. Procedural errors correspond to an error made when executing a single step of a recognised solution plan, whereas control errors reflect mistakes made in the supervision of the solution procedure.

Once a solution strategy has been chosen, it must be executed but following the solution plan may not be trivial, particularly if the goal changes during execution (for example, the solution plan *Expand and Factorise* commonly applied to the factorisation problem considered earlier (see Figure 3.8). This strategy entails the selection and planning phases being revisited several times during the *Execution* phase. The order of firing steps can also lead to changes in problem states and hence in the skills required to complete the solution. During the solution of a problem, short-term memory is constantly being overwritten, which can result in control errors (such as premature stopping). This overwriting increases the cognitive load and the retrieval of lots of small pieces of knowledge requires the student to constantly engage in the procedures for matching and retrieving items from long-term memory, which is very prone to errors. Thus inefficient solution techniques increase the cognitive load and are more likely to result in errors (see Figure 3.8).

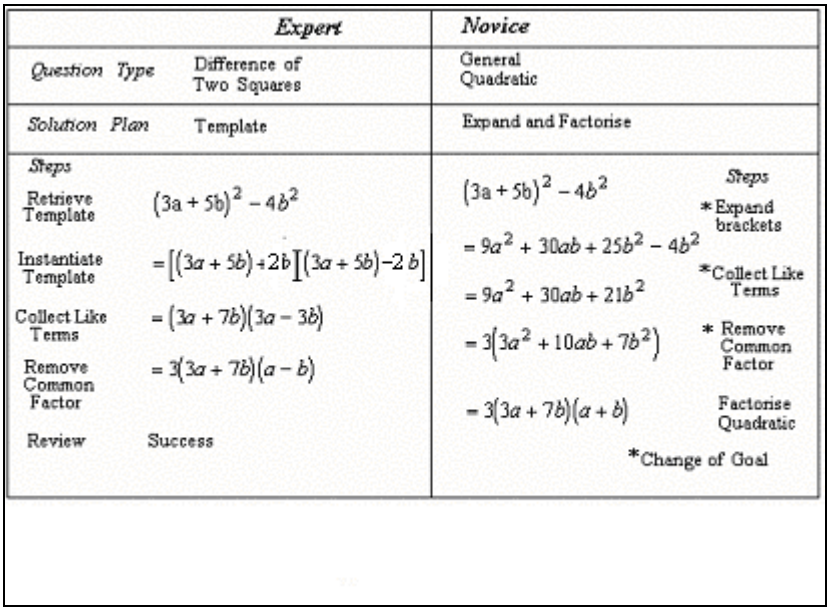

Figure 3-8 Effect of Expertise on Solution Strategy and Problem-solving Behaviour

A second reason for the variation in accuracy in problem solving of novices and experts is related to their different knowledge structures (Davis, 1984). Because experts have more critics, they are better at monitoring their progress and are less likely to lose their way (control errors) or to make slips (procedural errors). By reducing the number of steps required to solve a problem and by monitoring their own progress, experts minimise the chance of making errors. Self-monitoring comprises two main skills - look ahead and look back. By looking ahead, experts form an expectation of the new problem state that should emerge when a particular operation is performed; by looking back, they can assess the progress that they have made and determine the efficacy of the chosen strategy (Van Lehn, 1993). This is related to the final stage of problem solving (reviewing the results), which is now examined.

3.3.4 Reviewing results

A final characteristic that distinguishes experts from novices is the analysis that is undertaken at the *end* of the problem solving process. The purpose of reviewing the success of a strategy is to adapt the subject's knowledge structures. Just as they spend much time in analysing the problem at the start of the process, experts also spend time

in reviewing their results at the end of the process. Novices need to be encouraged to do this.

There are two main purposes of this stage of problem solving: to evaluate the correctness of the answer and to evaluate the efficiency of the chosen solution strategy. When novices do engage in this stage, they typically only perform the first function. Because their own checking rules can be erroneous, novices may compound their errors leading to degradation of their knowledge structures (Perrenet and Wolters, 1994). One purpose of our diagnostic system is to improve this stage of problem solving for novices, because this is how experts strengthen their knowledge structures (including their critics). Table 3.1 summarises the problem-solving behaviours demonstrated by experts and novices in the domain of mathematics.

Table 3-1 Expertise in Mathematical Problem Solving and its effect on behaviour

Problem Solving Stage	Output	Expert	Novice
Interpret	Question Type	What is this? Have I done this before?	What is this?
Plan	Solution Technique	How did I solve past problem? Can I reuse the solution? Can I adapt the solution?	Is there anything that I can do?
Execute	Answer	Look ahead - what do I expect to emerge from this step? Look back - is this correct so far?	Do It! Charge, Teddy!!

Review	Reflection on results	Did I answer the question? Is the answer correct? Why? Is this the only way to solve the problem? Is this the best way to solve the problem?	Thank God that's over!

In the next section, we describe the model of algebraic problem solving that we have developed and outline how it encapsulates the four stages of problem solving.

3.4 A Computational Model of Algebraic Problem Solving

We have adapted and extended Polya's four-stage model of general problem solving to develop a model that allows us to determine some of the conceptual activities undertaken by students during algebraic problem solving, not only the procedural ones. The model is now detailed.

3.4.1 Description of the Model

The identification of a student's question categorisation and choice of solution technique are the keys to effective error diagnosis (Ohlsson and Langley, 1988). Unless we know *what* a student is attempting to do, we cannot determine *why* they are attempting to do it and hence cannot realistically expect to address the fundamental causes of errors, viz. students' misconceptions. However, *what* a student is attempting to do varies from student to student and depends upon many factors including their level of mastery of important concepts. A model of an individual student's problem-solving behaviour must therefore infer the solution technique they were using.

The model of algebraic problem solving was derived from extensive analysis of the behaviour of many students in algebra tests (these are discussed in detail in chapter four). It proposes a computationally feasible method for each of Polya's phases as follows:

- *Interpreting* the problem: The selection and relative weighting of features relevant for a given question type are determined according to a student's knowledge which is represented as a discrimination network (see appendices). This is a classification task and the output from this stage is the value of the attribute *Question Type*.
- *Planning* a solution: Plans used by past students (who have interpreted the problem in the same way as the current student) are accessed. These are then ranked by generating the similarity scores between the current student's answer and those given in the past. This is a planning task and the output from this stage is the value of the attribute *Solution Technique*.
- *Executing* the plan: The accessed plans are adapted in terms of the current problem and then executed in order to determine which one most adequately describes the error(s) that the current student has made. The output from this stage is the *Student Answer*.
- *Reviewing* the process: Plans that successfully represent the current student's misconceptions are stored for later use along with an explanation of any errors. The output from this stage is the diagnosis.

The computational model is shown in Figure 3.9 and Figure 3.10 compares it with Polya's model. The remainder of this section details the operational phases of the model of algebraic problem-solving.

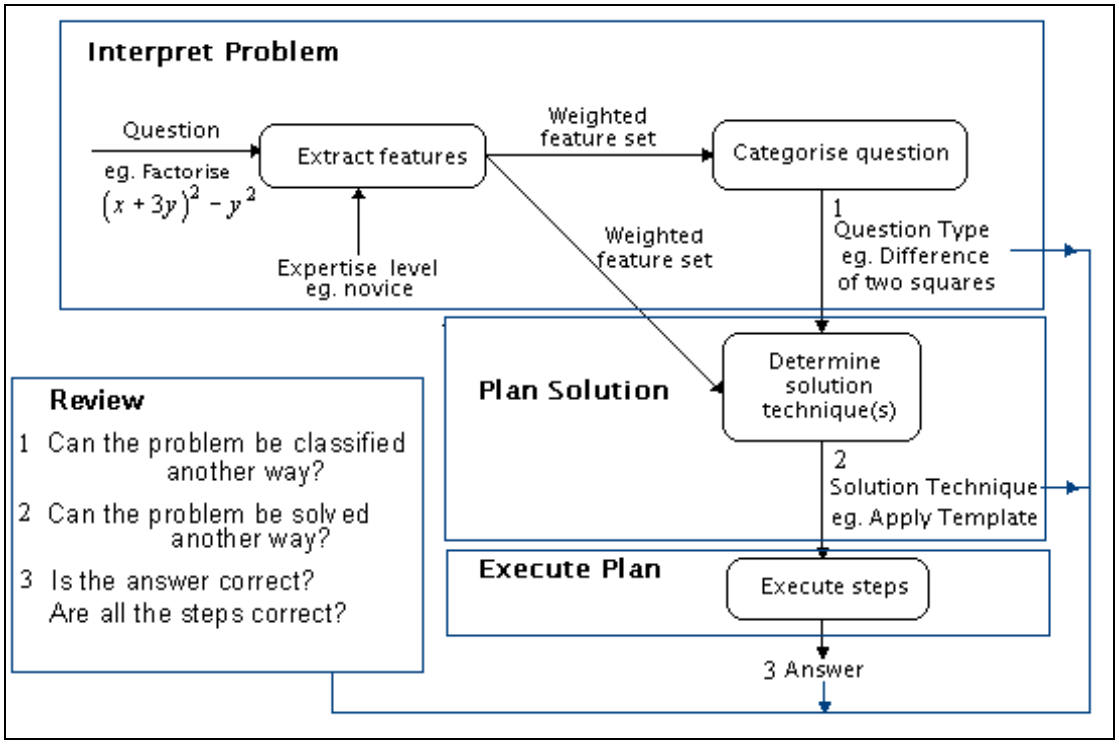

Figure 3-9 The Computational Model of Algebraic Problem Solving

Polya	Interpret the Question	Plan Solution	Execute the Plan	Review Results
Our Model	**Classify**	**Plan**	**Execute**	**Review**
Output	*Question Type*	*Solution Technique*	*Answer*	*Feedback*

Figure 3-10 Comparison of the Computational Model and Polya's Model

3.4.2 Problem Interpretation and Solution Planning

In this section, we examine how the related steps of Interpretation and Planning are represented in our model. Pattern recognition is fundamental to problem solving. Experts can recognise large, meaningful patterns whereas novices tend to focus on individual elements. This does not mean that experts have greater perceptual ability *per se*, but that their knowledge base is better organised than is that of a novice. This impacts on all stages of the problem-solving process. To commence solving a

problem, the student must form a representation of it. This representation is then compared with their existing knowledge structures to interpret what the problem is asking. To achieve full cognitive diagnosis, our system must therefore be able to represent a single question in different ways, because students at different levels of expertise will focus on different problem features and will impute different priorities to these features. In other words, we are attempting to recreate a student's representation of a problem that is functionally dependent upon their level of expertise and their measures of similarity.

What changes between different solvers on a given problem (or for an individual on a variety of problems) is the output from the categorisation and selection stages. For an expert with many schemata, the solution process is stable because it does not regress under cognitive load, nor does it vary with the problem details. Having recognised the problem type they instantaneously retrieve the appropriate solution technique because they perform problem categorisation on the basis of the moves required for solution (Sweller and Cooper, 1985). However a problem solver who is in the transition phase from novice to expert may categorise the same problem in different ways if details such as the degree of difficulty or order of presentation change, and hence apply different solution techniques.

We model the cognitive processes for similarity matching and retrieval by employing discrimination networks because they provide us with a means of representing these processes in a computationally feasible manner and they link question types to available solution techniques. Each network contains a set of ordered questions that the student must answer in order to determine both the category to which a particular problem belongs and an appropriate solution technique. Figure 3.11 shows three possible categorisations that pertain to the Difference of Two Squares problem *Factorise* $(x+3y)^2-y^2$ and the different techniques that can be applied to the problem. At one extreme is the expert approach (which results in the application of the appropriate template as the choice of solution technique), whilst at the other extreme is the novice approach (ie. means-end analysis). The other approach is Generalised

Distributivity, which represents an attempt to apply a technique that results from extrapolating an existing, but inappropriate, template.

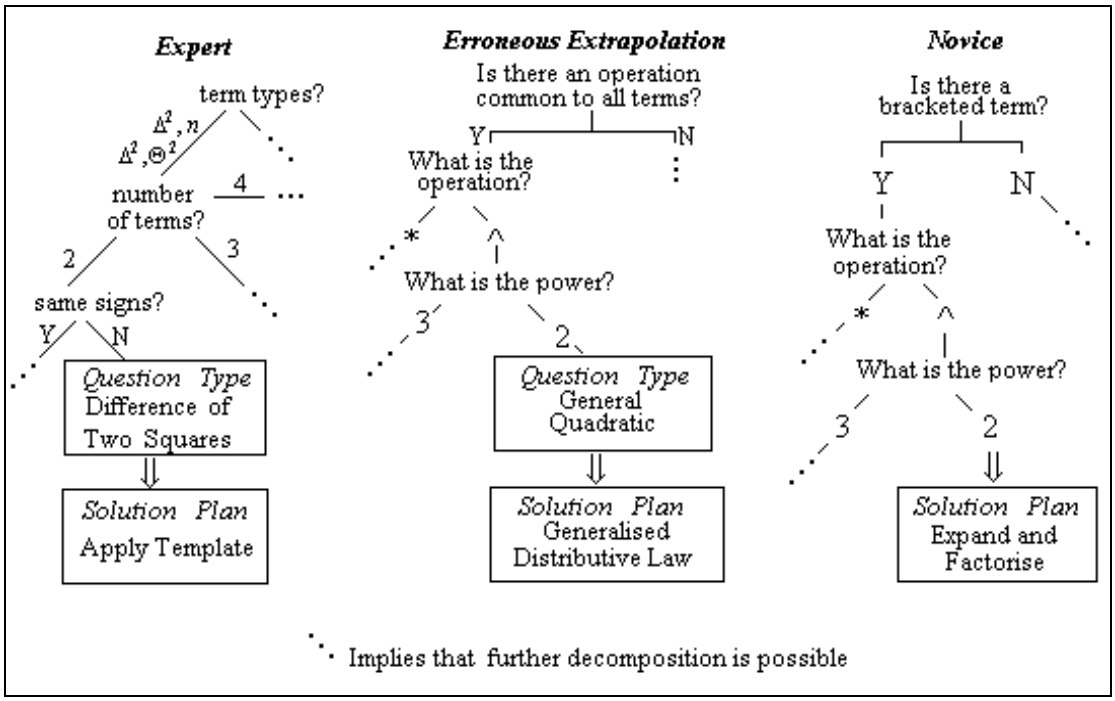

Figure 3-11 Problem Interpretation

This analysis indicates that, whilst a problem only belongs to a single category, it can possibly be solved by a variety of approaches. Therefore, a cognitive diagnostic system that is based on this model of algebraic problem solving must recognise all of the techniques that are commonly applied to a problem that belongs to a particular category. Since the different techniques give rise to very different answers, the system also requires methods for distinguishing between them (this point is the basis of the error analysis detailed in chapter four and of the design of the cognitive diagnostic system that is detailed in chapter five).

3.4.3 Execution of Plan

The first two stages of the problem solving model focus on the conceptual processes that a student undertakes when solving problems. The execution of the chosen solution technique, however, shows up the procedural and control errors that a student

makes. Cases in the *Solution Technique* library are plans that link each step of a particular plan to its associated set of common errors (malrules). These errors are well documented in the literature and have been used to construct the initial case library in the algebra diagnostic system.

Consider the problem *Factorise* $(x+3y)^2 - y^2$, which is recognised by an expert as a Difference of Two Squares problem (based on the fact that it contains two terms that are perfect squares and are opposite in signs). The optimal solution technique is to apply the template for this question type. The first discrimination network shown in Figure 3.11 represents this analysis. However, an early learner interpreting this problem may not recognise the form and, instead, is influenced by the fact that it contains a bracketed term which is to be squared, and so their first step in solving the question is to expand the bracketed term. This represents a shift in the goal (from *Factorise* to *Expand*) and so the student must now begin the solution process with the new goal and the reduced problem. Once this has been completed, the student should *Collect Like Terms* (new goal) and finally return to *Factorise* with the new problem. The third discrimination network shown in Figure 3.11 represents this. Because novices are keen to start solving a problem, they usually do not formulate a complete solution plan at the outset. Instead, as they perform each operation, they revise their progress and plan the next step. This continues until the student either reaches an impasse or perceives that they have completed the problem. Therefore, the system must compare the output at each of these points with the student's answer. The second approach in Figure 3.11 is an erroneous method that corresponds to an extrapolation error.

3.5 Summary

One way in which the effectiveness of teaching and learning can be gauged is by evaluating a student's performance in problem solving. Problem-solving performance changes as the subject spends more time engaged in problem solving constructs new

knowledge. Hence examining problem-solving performance provides information about how students learn, including the learner's concepts of the task and their ability to transfer their knowledge to the solution of novel problems. Solving routine algebraic problems requires the student to select a schema, instantiate it and then execute the associated solution strategy. Thus, algebraic problem solving requires the student to identify what the question is asking, to plan a solution and then to execute the plan.

To interpret an algebra problem and to plan a solution strategy, students use a form of analogical reasoning, by matching the current problem situation with a familiar one stored in memory. To access their stored knowledge, students form a mapping between the current problem and existing representations (or schemata) in long term memory. The mapping is achieved in two stages that are based upon the goal of the problem, the problem environment and different problem states achieved during the solution process. The first of the two stages is a parsing or categorisation problem and the second task is a selection task where the student must choose a solution technique form the set of available ones. Once the mapping has been created, the stored problem can then be recalled along with its solution, which is adapted to fit the new problem. Three main types of errors were identified, viz. *conceptual*, *executive* and *checking* errors. Conceptual errors in problem solving occur when the learner either applies an inappropriate solution technique or incorrectly adapts an existing one to fit the current problem. Executive errors are those errors made during the execution of a solution plan and have two main types: procedural (ie. an error made when executing a single step in a solution plan) and control (ie. errors in the order of executing steps or premature halting of a solution plan). The final type of errors are checking errors, which correspond to inappropriate methods for checking answers and which commonly represent misconceptions about what is meant by checking an answer, what is meant by the term "solution" and what is meant by the term "variable".

The identification of a student's question categorisation and choice of solution technique are the keys to effective cognitive diagnosis, which should be based on a model of problem-solving behaviour. Polya defined a general model of human

problem solving comprising four stages: interpreting the problem, planning a solution, executing the plan and reviewing the success of the plan. We have adapted and extended Polya's four-stage model of general problem solving to the case of algebra. This model allows us to determine (at least) some of the conceptual activities undertaken by students during algebraic problem solving. This chapter contains a description of the model of algebraic problem solving that we have developed and outlined how it encapsulates the four stages of problem solving. These steps and the outputs are:

1. Classify the problem to identify available solution techniques - output is a value for the attribute *Question Type*,
2. Plan a solution by choosing an available solution technique - output is the value of the attribute *Solution Technique*,
3. Execute the solution plan - output is the *Answer*, and
4. Review the success of the solution plan - output is the diagnosis.

The model was outlined in Figure 3-10, which is reproduced below.

Polya	Interpret the Question	Plan Solution	Execute the Plan	Review Results
Our Model	**Classify**	**Plan**	**Execute**	**Review**
Output	*Question Type*	*Solution Technique*	*Answer*	*Feedback*

Figure 3-12 Comparison of the Computational Model and Polya's Model

In the next chapter, we detail the error analysis that was conducted to develop a taxonomy of errors that could be used to underpin the design of the diagnostic system.

Chapter 4 Structural Analysis of Algebra Error Data

The previous chapter reviewed research results into the manner in which students solve algebra problems and the error-making process, and then presented the model of algebraic problem solving that was developed to serve as the basis of operation of a cognitive diagnostic system for algebra. Errors have been well documented and have been related to the processes of learning and problem solving (for example, Matz, 1980, 1982, Sleeman, 1984 and Payne and Squibb, 1990), but have not previously been classified in a manner that supports cognitive diagnosis beyond the level of identifying procedural errors.

What is missing from this body of work is a taxonomy of algebra errors that enables a computerised system to use a student's one-line answer to infer the solution technique that led to it. Because the chosen solution technique determines the skills required to solve a problem, it also determines the errors that are made and hence identifying the solution technique is the key to effective diagnosis (Ohlsson and Langley, 1988). As a result of this lack of an error taxonomy, there is a dearth of systems that are capable of accurately diagnosing algebra errors with the same level of success that has been achieved in the domain of procedural skills, eg. multi-digit subtraction. Instead, existing systems focus on determining errors in executing *procedural* skills, but do not attempt to analyse the *conceptual* activities that led to the student's choice of solution technique.

The current research addresses this issue by expanding the processes applied to achieve cognitive diagnosis, thereby improving the explanatory power of the resulting system. To achieve this, data were collected from hand-written solution protocols collected from the diagnostic tests administered to all students entering the Bachelor

of Engineering program at the University of Ballarat over a four-year period. Analysis of each student's working and the resulting one-line answer from a set of algebra problems was conducted to determine information about the cohort of students, particularly in terms of their preferences for particular solution techniques, their inconsistencies in problem solving and the factors that impact on these characteristics.

The main purpose of the error analysis was to develop a taxonomy of algebra errors that could be used to underpin a cognitive diagnostic system. However, the analysis also had the secondary purpose of attempting to uncover information that could be used to:

1. enrich case representation (by attempting to discover whether a hierarchy of solvability of problems existed, whether a measure of the degree of difficulty of a problem could be constructed and the set of factors that could be used to calculate such a measure),
2. direct system searching to improve efficiency (by attempting to discover patterns in an individual's choices of solution techniques as demonstrated by the migration of particular error types across questions that have different types, by attempting to evaluate a student's level of expertise, and by attempting to discover patterns in the entire group's choices of solution techniques), and
3. increase the explanatory power of the resulting system (to enable it to explain why students make mistakes during problem solving and the nature of the errors that they make by considering how conceptual, executive and checking errors are manifested, not simply procedural errors).

In this chapter, the analysis of error data is presented and the results are linked to the processes discussed in chapter three. The taxonomy of errors is detailed in the first section of this chapter, followed by a discussion of inconsistencies in a learner's problem solving performance and the factors that impact on an individual's choice of solution techniques.

4.1 Analysis of the Test Data at the University of Ballarat

The aims of mathematics teaching include helping students to acquire the mathematical knowledge, ways of thinking and confidence to use mathematical expressions, representations and technology to interpret information and to develop accurate and efficient problem solving protocols. However, when a student first performs at the expert level, we cannot expect that they will subsequently always perform at that level; rather, students demonstrate great inconsistency in problem-solving performance (Sleeman, 1984, Siegler, 1987a, 1987b, Anderson, 1990, Payne and Squibb, 1990, Sweller, 1992). Hence, what is of interest to the teacher is to note when and why a student's problem-solving behaviour breaks down. Diagnostic tests conducted at universities, whether they are administered by pen-and-paper or by a computerised system, usually have the dual purposes of determining the level of mathematics units in which students should enrol and what, if any, remediation they require. However, the diagnosis provided by these tests is neither well-explained nor fine-grained. That is, the tests identify *broad* areas of knowledge that the student has not mastered, but because they do not identify the solution technique that was adopted, they cannot achieve the required level of diagnosis that is required for student modelling within an interactive learning environment. Cognitive diagnosis needs much finer error analysis than that provided by such tests; in particular it requires that the student's solution protocols be identified.

Because our diagnostic system only has access to a question and a student's one-line answer to it, the system has to be capable of identifying what an algebra question is asking and of identifying all available solution techniques. These tasks require the system not simply to categorise the problem as a member of a broad family of related problems, but to use surface features extracted from the problem expression (such as the term types) because these can determine which techniques are available. As an example, consider the Difference of Two Squares problem "*Factorise* $(x+3y)^2 - y^2$". The problem expression contains a bracketed term that can be expanded as part of the solution technique **Expand and Factorise**, but this technique

is not relevant for a Difference of Two Squares problem such as "*Factorise* $16x^2 - 25y^2$". Hence classification of problems was conducted on the basis of the steps required for solution, which in turn is determined by the relevant structural features. This modelling was represented using redundant discrimination networks (see Chapter 3 and Appendix 2) and means that once the category to which a particular question belongs is known, so too is the set of available solution techniques.

For these reasons, we conducted deeper error analysis of the data collected from the diagnostic test at the University of Ballarat (this is similar to the test used at the University of Melbourne - see Swedosh, 1996). In this section, we provide details of the analysis that we undertook to determine the links between a student's single-line answer to an algebra question and the solution technique that led to it.

4.1.1 Analysis Of Student Responses

The initial analysis used the data from the ten questions that comprised the algebra section of the 1996 Diagnostic Test. At this point, the aim was to determine a method for decomposing a particular question into the set of skills that is required for correct solution (*cf.* Birenbaum *et al*, 1993). Therefore, each question was decomposed into its underlying attributes (the set of available solution techniques and the associated sets of skills) and all the errors made on each question were identified. Table 4-1 contains details of the attributes of each of the ten questions.

Table 4-1 Attributes of algebra questions from the 1996 Diagnostic Test at the University of Ballarat

Question Number	Question	Question Attributes
A1	Evaluate $\frac{2}{3}\left[-18+4\times 9-(6\times 8)\right]$	Order of operations, Directed number, Distributive Law.
A2	Write in the form a√2 + b√3 : $2\sqrt{2}+4\sqrt{18}+3\sqrt{48}+5\sqrt{3}$	Identifying factors, Square roots, Collecting like terms.

A3	Expand the following using positive indices only: $\left[\frac{-3b^3c^{-2}}{5a^3b^{-2}}\right]^3$	Converting terms with negative indices to equivalent form, Multiplying terms with a common base number, Raising a power to a power, Index Laws.
A4	Simplify the expression: $2\log_{10}26+\log_{10}75-\log_{10}3-2\log_{10}13$	Identifying factors and powers, Squaring numbers, Multiplication and Division of numbers, Log laws: $\log_a a=1$, Log of a product, Log of a quotient Log of a power term.
A5	Factorise: $(x+3y)^2-y^2$	Difference of two squares*[1], Collecting like terms (method 1) Expanding square term*, Factorisation* (method 2).
A6	Expand: $(x+y)^3$	Associative Law, Distributive Law, Collecting like terms (method 1) Pascal's triangle (method 2).
A7	Solve for a in terms of b: $e^a=3b$	ln as inverse of exp, Log laws: $\log_a a=1$, Log of power term.
A8	Solve for x: $x(x-5)=-6$	Distributive Law, Index Laws, Inverse operations (+ and -), Factorisation*
A9	Solve for x: $\frac{3}{x}-\frac{4}{a}=\frac{5}{b}$	Inverse operations (+ and - ; × and ÷), Inverting fractions, Finding common denominator*
A10	Solve for x and y: $x+3y=-6$ $x-3y=9$	Collecting like terms, Adding equations, Inverse operations (+ and - ; × and ÷), Directed number, Substitution, Multiplying fractions, Finding common denominator*

The purpose of this analysis was to determine whether the technique that a student used to answer a question could be uniquely identified from the final answer, because the diagnostic system would only have access to the student's one-line answer and not to all the intermediate steps that were taken. Success would mean that any misconceptions could then be identified and remediation structured accordingly. For problems that could be solved by a multiplicity of techniques, all of the available solution techniques and the associated skills were identified. Initially a schematic diagram for each solution technique that was applied to a particular problem was constructed; this identified all the skills required for solving the question and the errors that were associated with each skill. Examples of these diagrams for the

[1] * Indicates problems that can be solved by a multiplicity of methods.

Difference of Two Squares problem "*Factorise* $(x+3y)^2 - y^2$" are shown in Figures 4-1 and 4-2.

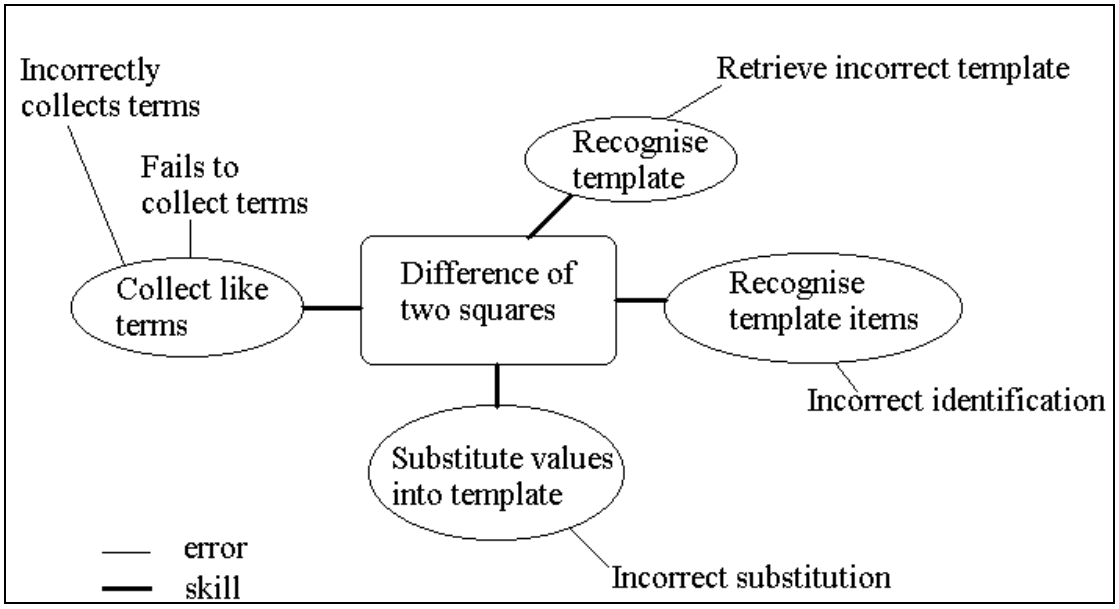

Figure 4-1 Map of skills and errors for applying the Template to a Difference of Two Squares problem

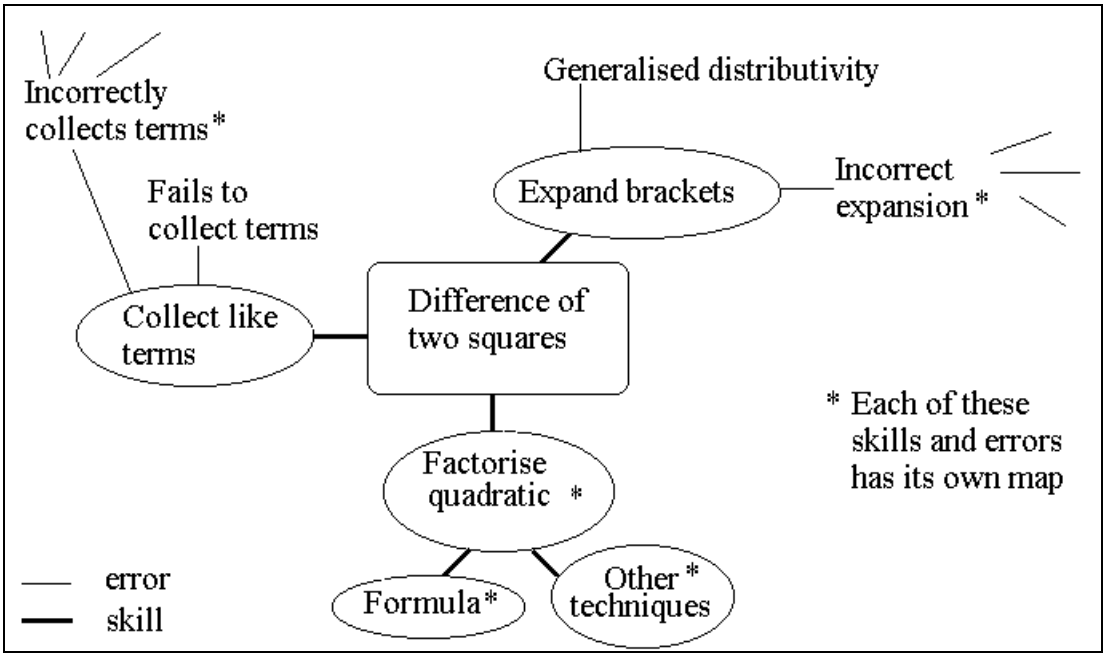

Figure 4-2 Map of skills and errors for the Expand and Factorise technique on a Difference of Two Squares problem with a bracketed term

Two sets of related questions were blocked for comparison. These sets were labelled **S1** which comprised questions A3 (index laws), A4 (log laws) and A7 (solving for the exponent), and **S2** which comprised questions A5 (factorising a quadratic expression), A6 (expanding a cubic) and A8 (solving a quadratic presented in non-standard form). Tables 4-2 and 4-3 contain details of the erroneous answers obtained for these sets of problems.

Table 4-2 Common erroneous answers for Set 1 Questions involving Indices and Logs

Terms in Final Answer	**Problem-solving Technique / Misconception**
Question A3 Expand the following using positive indices only: $\left[\frac{-3b^3c^{-2}}{5a^3b^{-2}}\right]^3$	
15 in denominator or 9 in numerator	Multiplied the number by the index
3375	Cubed each number but interpreted $\frac{1}{-125}$ *as* $\frac{125}{1}$
-5000 (for $(-5)^3$) in denominator	Confusion with standard notation
3 in numerator or -5 in denominator	Did not apply index to numerical term
b^9 in numerator and absence of c term	Ignored terms with negatives indices
b^{-6} in denominator or c^{-6} in numerator	Cannot remove negative indices
c^6 in numerator	Drops negative sign when converting to positive indices
a^6, b^5	Added indices when raising a power to a power
b^3 in numerator	Added indices when simplifying b^9 / b^{-6}
b^{19} in numerator or c^8 in denominator	Raised power to power
Question A4 Simplify the expression: $2\log_{10}26+\log_{10}75-\log_{10}3-2\log_{10}13$	
$\log_{10}85$	Used a generalised distributivity approach when combining log terms, various methods for dealing with coefficients
$\log_{10}72$, $\log_{10}13$	Used a generalised distributivity approach when combining log terms and grouped terms with same coefficient (common factor)
$\log_{10}13$, $\log_{10}3$	Used a generalised distributivity approach when combining log terms and "simplified" $\log_{10}72$ as $24.\log_{10}3$
$\log_{10}579$	Used log rules to simplify individual terms but then used a generalised distributivity approach when combining log terms
$2.\log_{10}1950$	Multiplied coefficients by extrapolating rule for adding logs
$2.\log_{10}2 + \log_{10}25$ or $\log_{10}(\frac{26^2\times75}{3\times13^2})$	Correct use of log rules but an incomplete form of the answer

Question A7 Solve for a in terms of b:	$e^a = 3b$
numerical or algebraic answer	Did not recognise the need for logs so employed algebraic manipulation
$\log_{10}(\ )$	Recognised the need for logs but used wrong base number
$\frac{1}{3}\log_e b$ *or* $3\log_e b$	Applied the appropriate logarithm, but treated as a linear operator

Table 4-3 Common erroneous answers for Set 2 Questions involving Expansion and Factorisation

Question A5 Factorise: $(x+3y)^2 - y^2$	
$(x + 2y)^2$	Used a generalised distributivity approach to whole problem
$x^2 + 8y^2$	Used a generalised distributivity approach when expanding the squared term
$x^2 + 2y^2$	Used a generalised distributivity approach when expanding the squared term AND expanded $(3y)^2$ as $3y^2$
$(x + 3y) + y + (x + 3y) - y$ $(x - 3)(y + 3)$ $x^2 - 10y^2$	Partial identification of template for the difference of two squares but applied incorrectly
$x^2 + 6xy + 8y^2$	Correctly expanded the square and collected like terms but did not attempt to factorise
$2y^2$	Expanded $(3y)^2$ as $3y^2$
$3xy$	Did not double the coefficient in the cross-product term when expanding the square
$9xy$	Squared the coefficient in the cross-product term when expanding the square
$18xy$	Squared and doubled the coefficient in the cross-product term when expanding the square
x^2y^2	Multiplied the pronumerals when collecting the like terms
$5y^2$	Treated $(3y)^2$ as $(2x3)y^2$
(..)(..)	Attempted some form of factorisation
Question A6 Expand: $(x+y)^3$	
$x^3 + y^3$	Used a generalised distributivity approach to whole problem
$(x + y)(..)$	Incomplete expansion
$x^3 + .. x^2y + ..xy^2 + y^3$	Correct form of terms but either incorrect template, arithmetic error or incomplete distribution
Terms where the sum of powers is not 3	Incomplete distribution or extrapolation of FOIL technique and lost track of the process
Question A8 Solve for x: $x(x-5) = -6$	

1 correct value	"Guess and Check". Did not recognise quadratic form and need for a second solution.
-1 and +6	Either solved $x^2 - 5x - 6 = 0$ or has predisposition to factors 1 and c without checking the consistency of the signs with the original problem
2 incorrect solutions	Used wrong formula
√-1	Either did not distribute in expanding $x(x - 5)$ or did not use correct order of operations and attempted to "undo" $(x - 5)$ without expanding
-6/5	Treated x^2 as $2x$ after first attempting to "undo" $(x - 5)$ without expanding
$x^2 - 5x = -6$	Failed to recognise form of the problem

The analysis led to a number of observations about students' choices of solution protocols:

1. Generalised distributivity errors were mainly associated with students at the lower end of the marking scale (although one student in the top 20% employed this extrapolation technique in the difference of two squares problem). Of the students who made such errors on algebra questions, 44% also made them on calculus questions (eg. for the function $f(x) = x^2 \cos x$, they produced the derivative $f'(x) = 2x \sin x$). Migration of such errors has implications for the extensibility of the diagnostic method presented here to student modelling in mathematical domains other than algebra.
2. For equation-solving problems, novice-level solution techniques such as "*Guess and Check*" and numerical substitution were associated with students at the lower end of the marking scale (ie. those with an overall score of less than 50%). This supports the expectation that novices are less accurate in problem solving than are experts.
3. Those students who were successful in expanding the cubic (A6) either used the template and Pascal's triangle to determine the coefficients of terms, or repeated application of the distributive law (which is a combination of the associative and distributive laws). All but one of the students who employed other heuristics (such as an extrapolation of the FOIL technique) lost track of the process and omitted terms. This provides evidence that solution techniques that can only be correctly applied to a limited set of problems should be avoided, unless students clearly understand the conditions under which they do and do not apply.

4. The misgeneralised Null Factor Law was not observed in the form which is reported in the literature. When solving the quadratic equation $x(x - 5) = -6$, students must search for the factors of -6 and some did show a natural preference for using the factors 1 and 6. However their working indicated that this was not due to the reported misgeneralisation of the Null Factor Law. If the latter were true, students would have written $x = -6$ or $x - 5 = -6$, which was not observed.
5. Within each of the two sets, no strict hierarchy of solvability existed. That is, success on one particular question within a set did not guarantee success on the other questions in the set nor did failure on one particular question within a set guarantee failure on the other questions in the set.

These observations replicated many of those that Matz had made (Matz, 1980, 1982). However, there were a number of shortcomings with the original diagnostic test. It only included one example of each question type, which prevented us from drawing any conclusions about the stability of an individual's problem solving performance or the factors that affect the choice of solution techniques. In light of these observations, a new diagnostic test that focused solely on algebra was developed and introduced in 1997. The new test included the ten algebra questions from the original test, plus an extra twenty-three related questions at different levels of difficulty (see Appendix 1 for the new test). As for the original test, all techniques that were employed for solving individual questions and the associated forms of the final answer were identified. Analysis of the results from the expanded test resulted in a number of new observations.

- Templates (expert approach) were more commonly used for expansion questions than for factorisation questions, where heuristics (novice level) were most commonly adopted. For example, student working indicated that students were generally comfortable when expanding a term such as $(x + y)^2$, but recognition of the factorised form of an expression such as $x^2 + 4x + 4$ was low.
- Expansion templates for exponentiation problems were more commonly used for problems involving squaring than for problems involving cubing. In the latter

case, students commonly produced the following line of working $(x+y)^3 = (x+y)(x^2+2xy+y^2)$ without any other steps. That is, they adopted a mixture of a novice-level approach (repeated application of the distributive law) and an expert-level approach (ie. the use of the template for expanding the squared term).

- Expansion templates for exponentiation problems were most commonly used when coefficients of all terms inside the brackets were 1, and expansion of perfect squares by generalised distributivity was less likely to be used when coefficients of all terms inside the brackets were 1. For example, generalised distributivity was not often applied to expanding the squared term $(x+y)^2$ in the cubic problem, but it was commonly used when expanding the term $(x+3y)^2$ in the Difference of Two Squares problem.
- The adoption of novice solution techniques such as numerical substitution for equation-solving problems were associated with students at the lower end of the marking scale.
- Problems involving the solution of quadratic equations were most commonly solved by search (novice level) or factorisation followed by the application of the Null Factor Law (intermediate level). Very few students (less than 3%) applied the quadratic formula (expert level) to the solution process.
- The highest baulking rates and lowest success rates were associated with questions where **recognition** is important (eg. problems involving logarithms and exponentials), and the lowest baulking rates and highest success rates were associated with questions which can be approached in a more procedural manner (eg. expansion problems). This is summarised in Table 4-4.
- Increasing the detail of a problem, without increasing its degree of difficulty, can induce a high baulking rate (eg. the question "Solve for *x*: $e^{3t}x = \frac{1}{3}e^{3t} + e^t + 2$"). The main type of error made here was attempting to apply logarithms, indicating that the subject was distracted by the presence of the exponential term on the left-hand side of the equation.

- No student ever showed written evidence of checking their answers, although some mental checking may have taken place. This indicated that students need assistance in developing and applying their own checking rules.

Table 4-4 Baulking Rates and Success Rates associated with different problem categories

Question Category	Baulking Rate	Success Rate
Expansion	Low (< 10%)	High (50%)
Factorisation	Medium (25 - 30%)	Medium for numerical (40-50%) Very low for algebraic (< 10%)
Logs, exponents	High (> 50%)	Low (10 - 25%)
Algebraic fractions	Medium (30%)	Very low (< 10%)

The success of the preliminary work indicated that diagnosis of errors and their causes could be expedited by using the problem environment to categorise problems and to determine available solution techniques (Mays *et al*, 1997). However, other desirable characteristics, such as a hierarchy of solvability, did not emerge. Because the test questions were presented in mathematical notation and in standard form, students were not required to formulate their own expressions and equations. Despite this, it is **not** true that only procedural skills were investigated; choosing an appropriate solution technique is itself a conceptual activity. This confirmed the decision to use the identification of solution techniques as the basis of diagnosis within our system.

Therefore, the relationship between the *form* of an answer and the technique that produced it was analysed. The analysis revealed that, for incorrect answers, we could correctly infer the solution technique employed from the single line answer. This was crucial because a completely automated system only has access to the student's single-line answer. Because questions are not hard-coded in our system, there is no predetermined link between a particular question and the student's answer. Instead the

system must be capable of determining the most probable solution path that students followed to arrive at their answers. The next section contains the details of this second tier of analysis.

4.1.2 Relationship Between Solving Technique And Final Answer

The purpose of the previous analysis was to determine whether a student's method of solution could be inferred from the structure of their final answer. The success of that analysis led to the next tier of analysis, viz. the identification of factors that could be used to develop a taxonomy of errors. For example, it had already been observed that the question *Factorise:* $(x+3y)^2 - y^2$ was solved by a variety of techniques (not all of them valid). Common incorrect responses to this question, the adopted solution technique and the associated values of expertise level and question category are shown in Table 4-5. (Note that the label "Intermediate" is applied to the Generalised Distributivity technique, because students must form a view of the *entire* problem to apply this technique to a factorisation problem. On the other hand, the means-end approach is labelled "Novice" because it implies that the student simply focused on one term in the expression and adopted a means-end approach to solving the problem. Whilst these labels may not be agreed to by all mathematics educators, it is true that these solution methods represent different interpretations by students and therefore require different treatments in the diagnostic system.)

Table 4-5 Relationship between a student's answer, their level of expertise and solution technique

Question	**Operational Expertise**	**Question Category**	**Solution Technique**
Factorise: $(x+3y)^2 - y^2$			
Correct Answer: (x + 2y)(x + 4y))	Expert	Difference of Two Squares	Template $a^2 - b^2 = (a + b)(a - b)$
(x + 3y+ y) + (x + 3y -y) (x + 4y) + (x + 2y)	Expert	Difference of Two Squares	Partial identification of template for the difference of two squares but applied incorrectly. Used the incorrect template $a^2 - b^2 = (a + b) + (a - b)$.

$(x - 3)(y + 3)$	Expert	Difference of Two Squares	Partial identification of template for the difference of two squares but applied incorrectly.
$(x + 2y)^2$	Intermediate	General Quadratic	Used a generalised distributivity approach to the entire problem. Factorised $a^2 - b^2$ as $(a - b)^2$.
$x^2 + 8y^2$	Novice/ Intermediate		Expanded the bracketed term using a generalised distributivity approach. Expanded $(a - b)^2$ as $a^2 - b^2$. Correctly expanded $(3y)^2$ as $9y^2$. Did not attempt to factorise - control error.
$x^2 + 2y^2$	Novice/ Intermediate		Expanded the bracketed term using a generalised distributivity approach. Expanded $(3y)^2$ as $3y^2$ Did not attempt to factorise - control error.
$x^2 + 6xy + 8y^2$	Novice		Correctly expanded the square and collected like terms but did not attempt to factorise. Control error.
$x^2 + 6xy + 2y^2$	Novice		Expanded the square and collected like terms. Expanded $(3y)^2$ as $3y^2$ Did not attempt to factorise - control error.
$x^2 + 6xy + 5y^2$	Novice		Expanded the square and collected like terms. Expanded $(3y)^2$ as $(2x3)y^2$ Did not attempt to factorise - control error.
$x^2 + 9xy + 8y^2$	Novice		Expanded the square and collected like terms. Multiplied the coefficients of the squared terms to find the coefficient of the cross-product term. Did not attempt to factorise - control error.
$x^2 + 18xy + 8y^2$	Novice		Expanded the square and collected like terms. Multiplied the coefficients of the squared terms and doubled the result to obtain the coefficient of the cross-product term. Did not attempt to factorise - control error.
$x^2 + 6x^2y^2 + 8y^2$	Novice		Expanded the square and collected like terms. Multiplied the pronumerals when collecting the like terms. Did not attempt to factorise - control error.

Having identified the different techniques and the different answers that they produce, the next step in the analysis was to group answers that were produced by variations of the same technique (see Table 4-6). The purpose here was to determine the similarities between the forms of answers that resulted from a single technique as well as the differences between the forms of answers that resulted from different techniques.

Table 4-6 Relationship between Solution Technique and the Form of an Answer for a Difference of Two Squares Factorisation Problem

Solution Technique	Terms in Erroneous Answers	Number of Terms in Answer	Term Types in Answer
Template	$(x+3y+y)+(x+3y-y)$ $(x+4y)+(x+2y)$ $(x-3)(y+3)$	2 1	linear -bracket, linear -bracket linear-cross-bracket
Generalised Distributivity	$(x+2y)^2$	1	quadratic-bracket
Expand and Factorise (expansion step is performed by template or repeated application of the distributive law)	x^2 one of $[+6xy, +3xy, +9xy, +18xy]$ one of $[+2y^2, +5y^2, +8y^2]$	3	quadratic-pronumeral, linear-cross-pronumeral, quadratic-pronumeral
Expand and Factorise (expansion step is performed by generalised distributivity)	x^2 one of $[+2y^2, +5y^2, +8y^2]$	2	quadratic-pronumeral, quadratic-pronumeral

Expand and Factorise (expansion step is performed by generalised distributivity, but collecting like terms involves the following malrule: $+ny^2 - y^2 \rightarrow n$)	x^2 one of [+3, +6, +9]	2	quadratic-pronumeral, numerical

Similar analysis of the answers to two other questions (*Expand* $(x+y)^3$ and *Expand* $(a-b)^2$) are summarised in Tables 4-7 and 4-8.

Table 4-7 Relationship of Solution Technique to the Answer Form for Expanding a Cubic

Solution Technique	Erroneous Answers	Number of Terms in Answer	Term Types in Answer
Template	$x^3 + x^2y + xy^2 + y^3$ $x^3 + 2x^2y + 2xy^2 + y^3$	4	cubic-pronumeral, cross-linear-pronumeral-quadratic-pronumeral, cross-linear-pronumeral-quadratic-pronumeral, cubic-pronumeral
Generalised Distributivity	$x^3 + y^3$	2	cubic-pronumeral, cubic-pronumeral
Repeated Application of the Distributive Law	$x^3 + \ldots + y^3$	4 or more	These vary, but the types are incorrect (ie. the sum of powers is not 3).
Repeated Application of the Distributive Law	$(x+y)(x^2 + \ldots xy + y^2)$ NB. The value of the coefficient of the *xy* term in the second bracket varies, usually 1 or 2.	1	cross-linear-bracket-quadratic-bracket

Table 4-8 Relationship of Solution Technique to the Answer Form for Expanding a Square

Solution Technique	Erroneous Answers	Number of Terms in Answer	Term Types in Answer
Template	$a^2 - 2ab - b^2$ $a^2 + 2ab - b^2$	3	quadratic-pronumeral, linear-cross-pronumeral, quadratic-pronumeral
Generalised Distributivity	$a^2 - b^2$	2	quadratic-pronumeral, quadratic-pronumeral
Repeated Application of the Distributive Law	$a^2 - ab - ba - b^2$	4	quadratic-pronumeral, linear-cross-pronumeral, linear-cross-pronumeral, quadratic-pronumeral
Repeated Application of the Distributive Law	$(a-b)(a-b)$ $(a-b)(a+b)$	1	linear- cross-bracket

These observations led to the conclusion that the two most important features for deducing the solution technique from an answer are the *number* of terms in the answer and the *types* of terms in the answer. These features can be used to distinguish between the different solution techniques. For a particular solution technique, secondary features such as the values of coefficients and the signs on the terms in the answer can be used to distinguish between the different variations that can arise. For example, in Table 4-6 students who applied the solution technique "Expand and Factorise", where the expansion step was performed either by template or repeated application of the distributive law, produced a variety of answers depending upon the errors made for the expansion subproblem. For example, the coefficient of the cross-product term typically took one of the following values 3, 6, 9 or 18, whereas the coefficient of the *y*-squared term typically took one of the following values 2, 5 or 8. However, all of the answers had the same **structure** (in that they contained three terms, two of which were quadratic pronumeral terms and one of which was the linear

cross-product of two pronumerals). This meant that matching a student's answer on a particular problem with answers obtained in the past could be performed by using answer structure as the guide, and then having some means of distinguishing between the different variations. The decision taken at this point was to include a generative mechanism for each solution technique that was to be stored, and that these mechanisms should be modular in nature. For the current example, this means that the system should use the structure of the student's answer to identify the solution technique used for the complete problem as "Expand and Factorise", and surface features (ie. the actual values of the coefficients of the terms) to identify the method used to perform the expansion subproblem. This is discussed in detail in chapter six.

In summary, no pattern of behaviour pertinent to all students was observed. Instead, problem-solving behaviour broke down at different points for individual students; meaning that there is no "master" pattern in behaviour and so modelling must be done on an individual basis. In particular, it was found that it cannot be expected that a student will always operate at the same level of expertise; this point is addressed in more detail in section 4.3. In the next section, the results of the statistical analyses conducted on the diagnostic test data are discussed.

4.1.3 Statistical Analyses

Extensive analysis of students' problem-solving protocols on a variety of algebraic tasks was conducted. The aim of this analysis was to determine if there were factors that influence either an individual's problem-solving performance or the degree of difficulty for an individual question. The existence of such factors could then be incorporated into the diagnostic system to improve both its searching efficiency and its diagnostic accuracy. The data were collected over a four-year period from the expanded diagnostic test given to students entering the Bachelor of Engineering program at the University of Ballarat. The complete data set comprised 143 tests. In this section, the results of the statistical analyses that were conducted on the diagnostic test data, the entry level scores (TER) and students' Year 12 mathematics results are detailed.

4.1.3.1 Correlations between Diagnostic Test Scores, Maths Scores and Entry Scores

Students ranged in ages and ability levels, but the majority of students (about 75%) had completed their Year 12 studies in the year prior to entering the course. The remaining students had completed their secondary studies between 3 and 15 years prior to entering the course. For students in the former group, the first statistical tests that were conducted were used to test for the presence of a significant correlation between students' diagnostic test scores and either their Year 12 mathematics scores or their overall tertiary entry scores (TER). The existence of a significant correlation would mean that we could use either a student's TER score or their Year 12 mathematics score as a first predictor of their expertise level, which in turn could serve to predict their choice of solution techniques.

As was expected, it was found that student performance on the diagnostic test was significantly linearly correlated with their overall tertiary entry score (R-squared = 0.42, $t = 4.3$, $p < 0.0001$), and with their Year 12 mathematics scores (R-squared = 0.36, $t = 3.3$, $p = 0.003$). The scattergrams for the data are shown in Figures 4.3 and 4.4. From these results, it was concluded that either of these scores could be used to provide an initial estimate of a student's operational expertise and hence to predict the solution technique that they would most probably apply to a given problem.

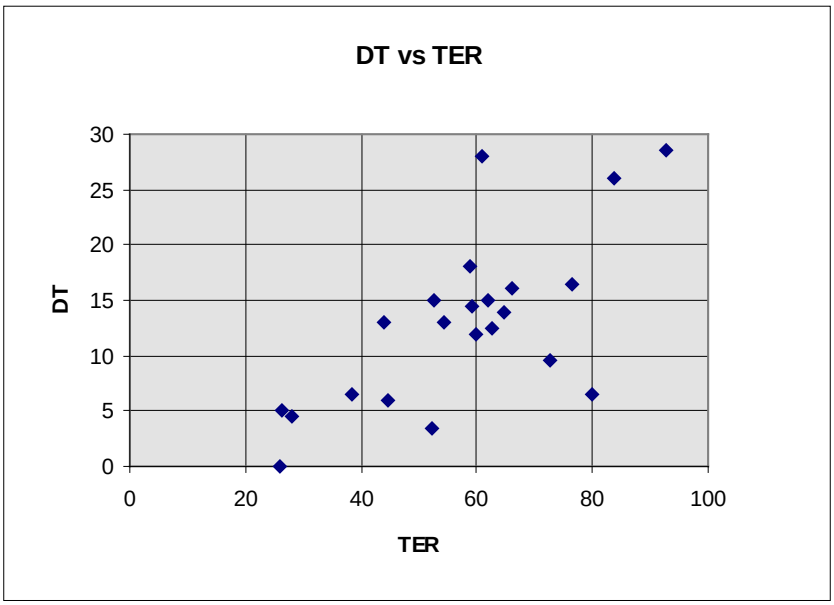

Figure 4-3 Relationship between diagnostic test scores and TER scores.

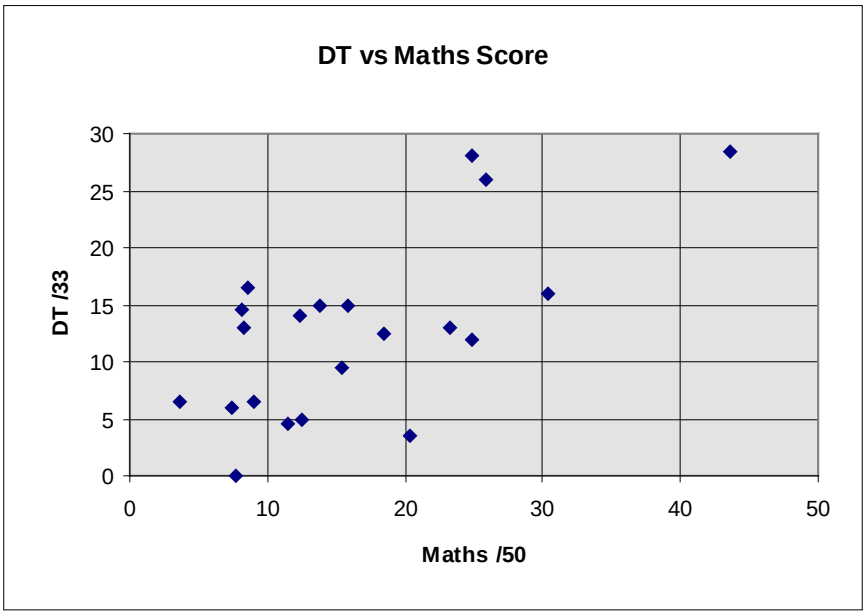

Figure 4-4 Relationship between Diagnostic Test scores and Mathematics scores.

For the group of students who had not studied mathematics for one or more years, we tested whether there was a significant correlation between the students' diagnostic test scores and the length of time since they had last studied mathematics. No such correlation was found. However the data were severely skewed and exhibited a negative trend; coupled with these observations we found that, with only one

exception, none of this cohort of students scored greater than 35% on the diagnostic test. This led us to conclude that for students who had not studied any mathematics for three or more years, we could assume that they would most probably operate at the novice level. That is, information such as the length of time since the student last studied mathematics, their tertiary entry score, or their score on the highest level mathematics subject that they have completed, provides reasonable starting points for assessing a student's expertise level. This information is similar to that recorded by the first generation of student models (Holt *et al*, 1994) and its purpose was to provide the diagnostic system with a starting point for the search during the diagnostic phase to improve efficiency.

4.1.3.2 Statistical Analyses of Question Data

Error analysis from the diagnostic tests provided us with "second-generation" information (that is, extended information about an individual's idiosyncrasies) and the test content. In particular, we determined that we could use incorrect responses to questions to infer the solution processes employed (see sections 4.2.1 and 4.2.2), but we did not observe a strict hierarchy either of solvability or of problem-solving approaches (this latter point is addressed in more detail in the remainder of this chapter). The remainder of the analyses detailed here focused on the relationships between questions and are based on the reduced set of 103 complete algebra tests from years 1998 and 1999. The purpose was two-fold: to identify factors that could be used to evaluate the degree of difficulty of a question and to identify factors that point to the solution techniques that would most probably be applied to a given question.

Table 4-9 Baulking, Error and Success Rates for each question on the Diagnostic Test

Question	Baulking Rate %	Success Rate %	Error Rate %
1	6	54	40
2	63	13	24
3	58	7	35
4	55	8	37
5	27	9	64
6	10	38	52
7	51	27	22
8	22	23	55
9	49	11	40
10	31	30	39
11	31	25	44
12	39	11	50
13	9	50	41
14	39	19	42
15	38	12	50
16	33	11	56
17	31	33	36
18	41	34	25
19	28	32	40
20	33	10	57
21	15	50	35
22	24	2	74
23	60	7	33
24	30	9	61
25	10	72	18
26	29	19	52
27	34	13	53
28	44	16	40
29	30	49	21
30	76	8	16
31	49	14	37
32	54	46	0
33	32	8	60

The first data that were examined were the baulking, success and error rates for each question on the test (see Tables 4-9 to 4-11 and Figures 4-5 to 4-8). As expected, the baulking and success rates were significantly, negatively correlated (R-Squared =

0.328, *t*-value -3.89339, *p*-value 0.0005). However, the correlation between baulking and error rates was not as strong as expected (R-Squared = 0.178, *t*-value -2.676, *p*-value 0.05). This indicates that a measure of the degree of difficulty of a question would require the use of factors other than simply the baulking rate.

Table 4-10 r-squared and t values for correlations between Baulking, Success and Error Rates

	BAULKING	SUCCESS
SUCCESS	0.329 (*t* = -4.02423)	
ERROR	0.178 (*t* = -2.676)	0.250 (*t* = -3.31797)

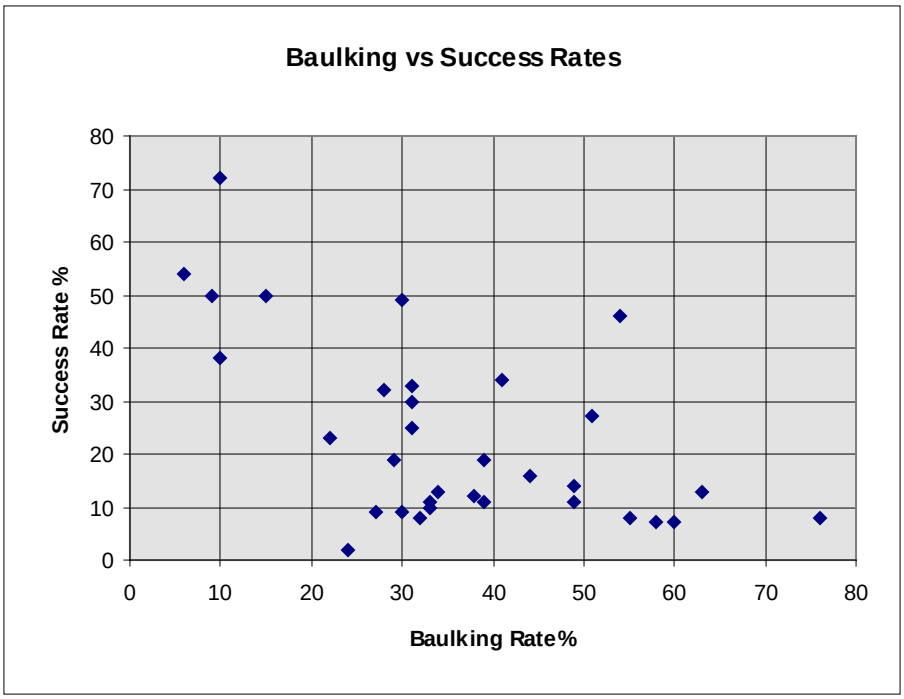

Figure 4-5 Scattergram of Baulking and Success Rates for each question

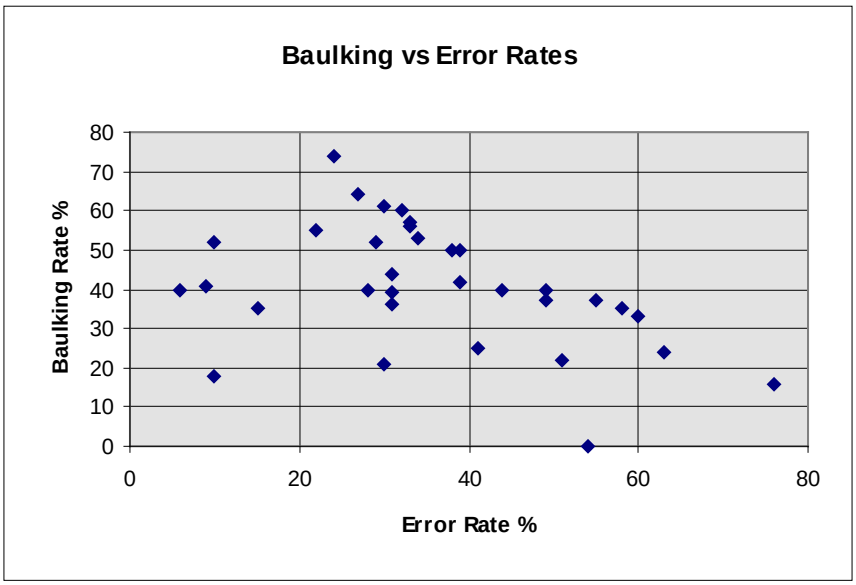

Figure 4-6 Scattergram of Baulking and Error Rates for each question

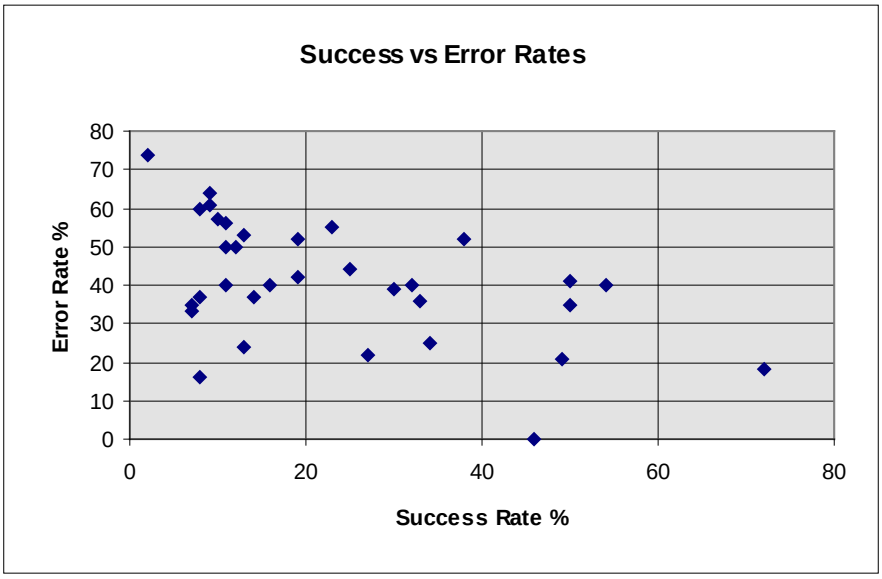

Figure 4-7 Scattergram of Success and Error Rates for each question

Due to the lack of clear information regarding the difficulty of questions, we then grouped related questions and reexamined the data (see Table 4-11). However, the analysis again failed to reveal the required information about hierarchies of solvability. That is, we discovered that a question's category on its own is insufficient to determine its degree of difficulty.

Table 4-11 Error, Success and Baulking Rates for Related Questions

QUESTION TYPE	QUESTION NUMBER	BAULKING RATE %	SUCCESS RATE %	ERROR RATE %
Evaluate	1	6	54	40
Surds	12	39	11	50
	2	63	13	24
Indices	3	58	7	35
	7	51	27	22
	17	31	33	36
	18	41	34	25
	19	28	32	40
	22	24	2	74
	23	60	7	33
Logs	32	54	46	0
	14	39	19	42
	4	55	8	37
Factors &	5	27	9	64
Expansion	13	9	50	41
	12	39	11	50
	6	10	38	52
	29	30	49	21
Solve	8	22	23	55
	26	29	19	52
	30	76	8	16
	31	49	14	37
	33	32	8	60
	10	31	30	39
Inequality	28	44	16	40
Fractions	11	31	25	44
	16	33	11	56
	9	49	11	40
	20	33	10	57
	21	15	50	35
	24	30	9	61
	25	10	72	18
	27	34	13	53

The statistical analysis provided information about initial estimates for a student's expertise level, but could not identify points at which a student might vary in their behaviour. It also failed to reveal a hierarchy of question solvability or a pattern of behaviour pertinent to all students. Thus the ideas of using measures of a student's expertise and a question's degree of difficulty to direct the searching phase were abandoned. Instead, the focus of the analysis was transferred to the identification of information that could improve the explanatory power of the system. Therefore, we reviewed each student's working for all questions on the diagnostic test to determine which other information should be incorporated into the system to improve its diagnostic capability. In particular, we examined the solution techniques applied to each question and the factors that influenced each student's choices. The results of these analyses are detailed in the remainder of this chapter.

4.2 Expertise and Inconsistencies in Problem Solving

In section 4.1, we demonstrated that a student's solution technique for a particular algebra problem could be deduced from the form of the resultant answer with a reasonable level of certainty. However, we also indicated that student performance is unstable; that is, we cannot expect that students will always apply the same technique to problems of the same type. The assumption of stability of problem-solving performance underpins a number of diagnostic systems, but, in this section, we provide the reasons why we avoided making it in the design of our system.

Firstly, we make a distinction between *conceptual*, *procedural* and *control* errors. The first error type appears to arise from a basic misunderstanding of algebra, and is more likely to be observed with problems where recognition is critical in choosing an appropriate solution technique. For example, the question "Solve for a: $e^a = b$" cannot be correctly answered unless the student recognises the form of the problem and the need to use logarithms to solve. In this problem, students may attempt to proceed by using the wrong base for the logarithms (which constitutes a procedural error), or by

employing solution techniques such as algebraic manipulation or arithmetic substitution (both of which are conceptual errors). However if a student fails to recognise the form of the problem *Factorise* $(x+3y)^2 - y^2$ as being a Difference of Two Squares, they can still proceed by adopting another (sub-optimal) solution technique.

The second error type refers to the case where a correct solution procedure has been applied to a problem, but one or more steps have either been omitted or incorrectly adapted. The third type of error (control errors) refers to those where the student either halts the solution process prematurely or otherwise loses track of the process (Matz, 1980). These errors are commonly encountered and so correct, partial answers are treated as errors in the diagnostic system. As an example, consider the factorisation question above. The most common solution technique applied to this question is *Expand and Factorise* and the corresponding solution plan contains three main steps: 1) expand the bracketed term, 2) collect like terms, and 3) factorise the resultant expression. However, the most common answer to the problem is $x^2 + 6xy + 8y^2$, which indicates that the student has either omitted the third step of the solution plan or cannot categorise the sub-problem that has emerged at this stage.

Initially analysis of solution protocols was only performed at the top level. If we deduced that a student had employed repeated application of the distributive law to expand a cubic, we did not look at how they subsequently expanded the squared term which emerges during solution. Since our aim is to improve student modelling by incorporating information about inconsistency during problem solving, we re-examined the diagnostic test data. In particular, we scrutinised students' problem solving protocols in much greater detail to detect shifts in operational expertise either across the entire test or on a single problem.

One of the most interesting findings was that expertise level for an entire test (judged from the solution techniques applied) was not entirely consistent either within or across individuals, even for students who were ranked in the top 10%. What we did

discover about members of this subgroup was that they always worked at either expert or intermediate level and, with only one exception, were never found to work at the novice level. Similarly, the lowest level (novice) solution techniques were always associated with students who ranked in the lower half of the cohort with only one exception. Therefore, a student's overall score on an algebra test provides a useful first estimate of their operational level of expertise. We also observed that for multi-stage problems, students who began solving at either the novice or intermediate level sometimes demonstrated shifts in behaviour when a change of goal emerged, whereas students who commenced solving at the expert level did not change their behaviour during the solution process.

Two examples from the diagnostic test are presented to illustrate this shift in operational expertise. The questions were **Factorise** $(x + 3y)^2 - y^2$ and **Expand** $(x+ y)^3$. For both questions it was found that, of the three possible approaches, the majority of students favoured the novice level (65% and 70% respectively). Although, the factorisation question could be approached in a variety of ways, the most common solution technique was a novice-level approach comprising the steps: expand the bracketed term, collect like terms and factorise the resultant expression. Similarly, the expansion question was most commonly solved by repeated application of the distributive law (novice level). Neither the intermediate nor expert approach to solving these problems exposes the student to a change in goal during the solution process. However, the novice approach to solving the factorisation problem is labour-intensive and involves several changes of goal. This means that the cognitive load for this approach is much higher than for the other two because the student has many manipulative processes to perform and must keep track of these processes. As a result, a student's problem-solving behaviour can degenerate (that is, the operational expertise level can decline). The questions and the observed problem solving protocols are detailed in Table 4-12.

Table 4-12 Solution protocols for two related problems

Factorise $(x + 3y)^2 - y^2$	**Expand** $(x + y)^3$
Novice: Expand and Factorise Sub-problem: Expand $(x + 3y)^2$ [2]Sub-problem: Factorise $x^2 + mxy + ny^2$	Novice: Repeated Distributive Law $(x + y)(x + y)^2$ Sub-problem: Expand $(x + y)^2$
Intermediate: Generalised Distributivity $(x + 3y – y)^2 = (x + 2y)^2$	Intermediate: Generalised Distributivity $x^3 + y^3$
Expert: Template $(x + 3y - y)(x + 3y + y) = (x + 2y)(x + 4y)$	Expert: Template $x^3 + 3x^2y + 3xy^2 + y^3$

The novice approach to these two problems results in the emergence of a sub-problem that involves expanding a perfect square. For each of these sub-problems, we observed shifts in behaviour (see Table 4-13), but there was no uniformity in the shifts. That is, these students started solving the original problem at the novice level, but all three levels of expertise were observed during the solution of the emergent sub-problem. Similar shifts were observed on other questions.

Table 4-13 Frequencies for solution techniques when expanding a perfect square

Expand $(x + 3y)^2$	**Expand** $(x + y)^2$
Novice: Repeated Distributive Law $(x + 3y)(x + 3y)$ Frequency: 65%	Novice: Repeated Distributive Law $(x + y)(x + y)$ Frequency: 35%
Intermediate: Generalised Distributivity $x^2 + (3y)^2$ Frequency: 10%	Intermediate: Generalised Distributivity $x^2 + y^2$ Frequency: 0%
Expert: Template $x^2 + 2.(x)(3y) + (3y)^2$ Frequency: 25%	Expert: Template $x^2 + 2.(x)(y) + (y)^2$ Frequency: 65%

[2] The values of *m* and *n* will vary, and depend upon the student's mastery of the index laws.

This analysis indicates that as the detail increases for a problem of a particular type, students who have operated at a high level of expertise often revert to more primitive solution processes. Students are more comfortable applying solution templates when all terms have unitary coefficients - ie. when there is an increase in the surface similarity between the given problem and solution templates as presented in textbooks.

4.3 Factors Affecting the Choice of Solution Technique

As demonstrated in chapter three, a learner's success in problem solving is determined, in part, by their choices of solution techniques. Questions on the revised diagnostic test that were handled with the greatest degree of success were those that were largely procedural (such as evaluation and expansion problems) and those that required little, if any, recognition to find an appropriate solution technique. Overall, high baulking rates and low success rates were associated with high recognition tasks, or simple tasks with a high degree of detail. On the other hand, problems that were largely procedural generated low baulking rates and higher success rates. The baulking and success rates for each question on the test for each of the years 1998 and 1999 are shown in Figures 4-8 and 4-9.

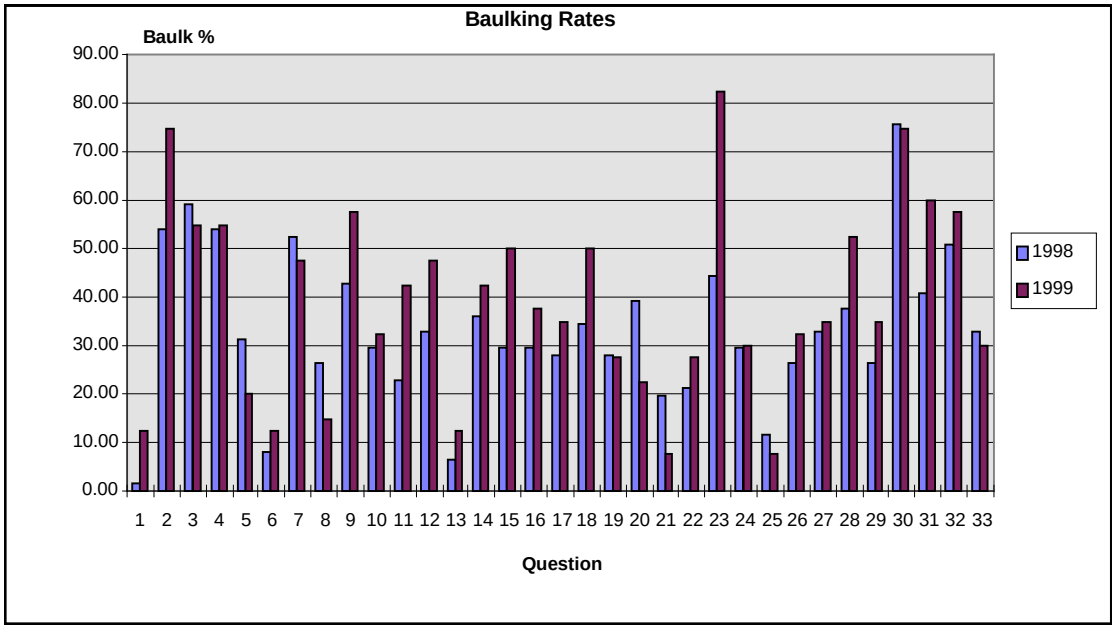

Figure 4-8 Baulking Rates for each Question on the Diagnostic Test (1998, 1999)

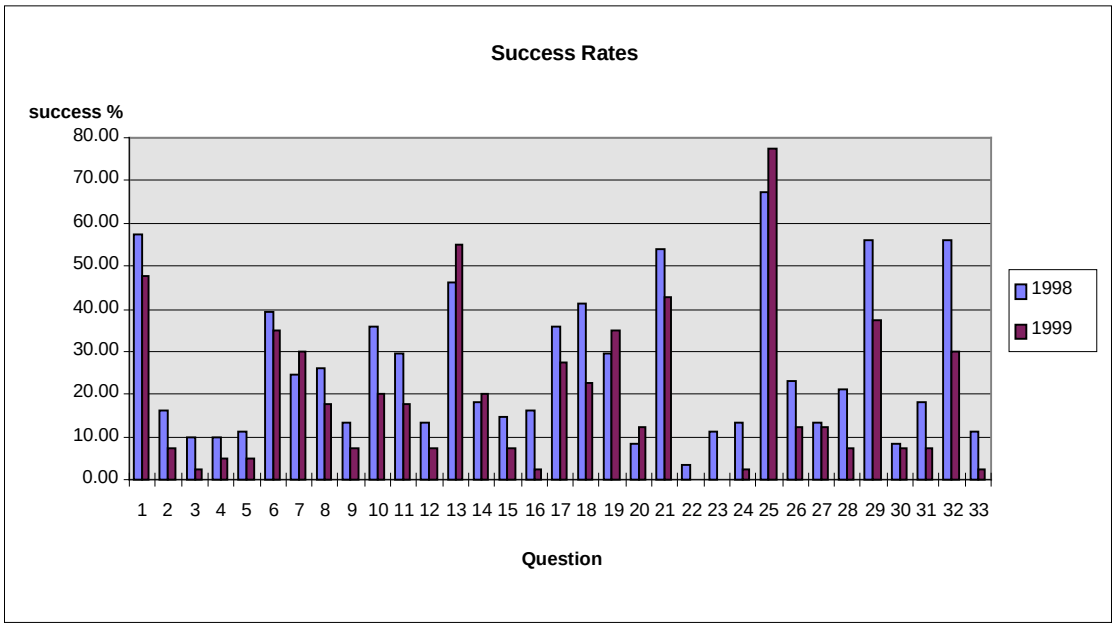

Figure 4-9 Success Rates for each Question on the Diagnostic Test (1998, 1999)

The combined baulking and success rates are shown in Figure 4-10.

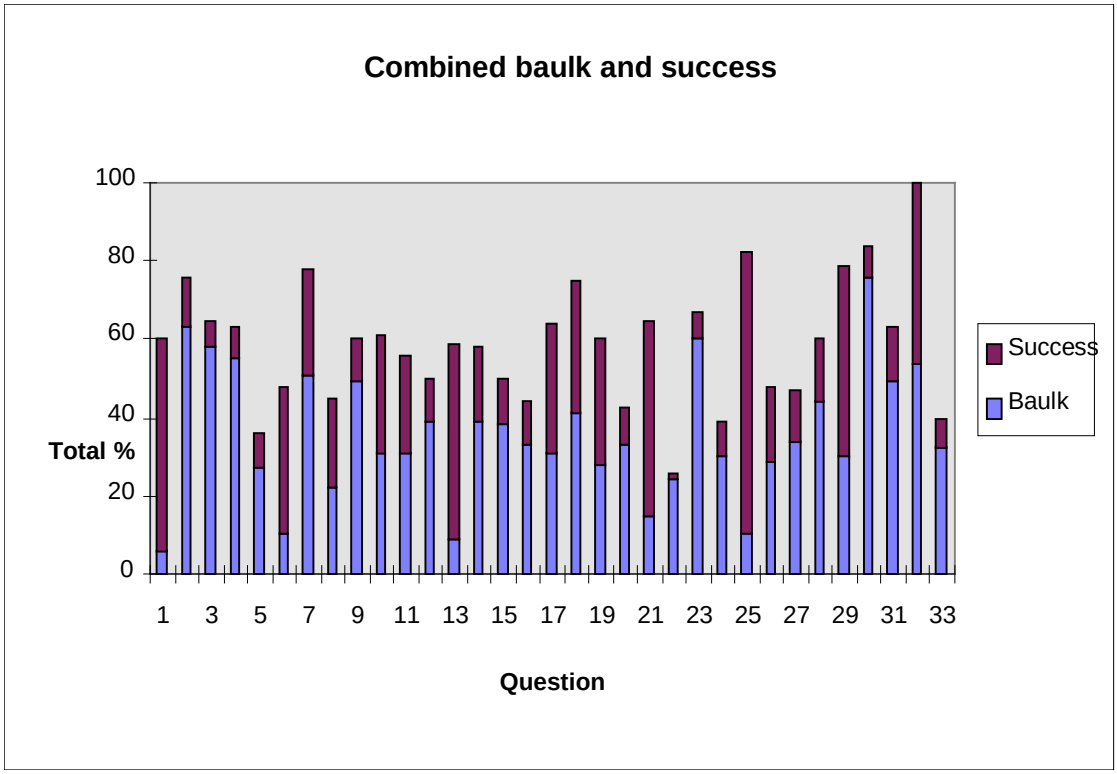

Figure 4-10 Combined Baulking and Success Rates for each Question on the Diagnostic Test (1998, 1999)

Analysis of the students' working on the diagnostic test revealed that an individual's choice of solution techniques is unstable and is affected by a number of *contextual* factors including the amount of recognition required to categorise a problem and the degree of detail of the problem, and *affective* factors such as a student's level of confidence. The impact of these factors is now examined in terms of the baulking and success rates.

4.3.1 Degree of Recognition Required for Solving a Question

For problems that require the student to recognise the form of the problem and the operations required for solving proved to be the most difficult questions; that is, they induced high baulking rates and produced low success rates. The most common problem types in this category were those involving logarithms or indices.

The most notable problem was question 32, viz. "*Solve* for *x*: $\log_a x = b$", which had a high baulking rate (54%) but which was answered correctly by all students who did attempt it. This is a trivial problem *provided* that the student recognises its form; that is once the student identifies what the problem is asking, it can be solved in a single step. This is an example of an *insight* problem (Van Lehn, 1989). Similar results were observed for question 7, viz. "*Solve* for *a*: $e^a = 3b$". This question also had a high baulking rate (51%), while over half of those students who did attempt the problem were successful. The remaining students who attempted the problem either made linearity errors (eg. $\ln(3b) \rightarrow 3\ln(b)$) or failed to correctly recognise the form of the problem and attempted to use some form of algebraic manipulation. Other problems involving logarithms (questions 4 and 14) also had high baulking rates (55% and 39% respectively) and low success rates (8% and 19% respectively). The most common errors were again linearity errors. These observations are summarised in Table 4-14.

Table 4-14 The Effect of Recognition on the Choice of Solution Technique

Question	Baulking Rate	Success Rate	Common Errors
32 Solve for x: $\log_a x = b$	54%	46%	None
7 Solve for a: $e^a = 3b$	51%	27%	Algebraic manipulation
4 Simplify $2\log_{10} 26 + \log_{10} 75 - \log_{10} 3 - 2\log_{10} 13$	55%	8%	Linearity
14 Simplify $\log_{10} 8 + \log_{10} 5 - \log_{10} 2$	39%	19%	Linearity
23 Write in simplest form $a\sqrt{a}$	60%	7%	$\sqrt{a^2} = a$ or a^3

The above results indicate that students have difficulty in recognising the form of problems involving logarithms and indices, even for very straightforward problems. The characteristics of expertise that have the most impact on recognition are: 1) experts *see* more than do novices (ie. their knowledge is greater and "chunking" is better), 2) experts adopt a top-down approach to problem solving whereas novices focus on small parts of a problem, 3) experts tend to focus on relational features of a problem whereas novices focus on superficial features, and 4) experts are better at recognising meaningful patterns than are novices. Research findings have shown that, whilst students use a form of analogical reasoning to determine solution techniques available for a given problem, they initially focus on the surface similarities between problems rather than relational similarities (for example, Gentner, 1988, 1989, Reed, 1989, Pierce and Gholson, 1994 and English and Sharry, 1996). To improve recognition, students need practice in *conceptual* activities (such as problem classification) rather than simply practising *procedural* tasks (similar to the work conducted by English and Sharry, 1996). This is fundamental to success in mathematics, because the skills of generalisation and abstraction are the keys to mathematical thinking.

4.3.2 Degree of Detail of a Question

The next factor that we considered was the degree of detail in a problem. The most outstanding problem in this regard was question 30, viz. "*Solve* for *x*: $e^{3t}x = \frac{1}{3}e^{3t} + e^{t} + 2$". This problem is another example of an insight problem, because it can be solved in a single step once the student recognises that the problem has the form $a.x = b$ and can be solved simply by dividing all terms by the coefficient of the *x* term. This problem had the highest baulking rate of all problems (76%) and a very low success rate (8%). The most common error type made on this problem was the application of natural logarithms, indicating that students failed to recognise the form of the problem.

The effect of problem detail was also observed in the two related questions 5 and 29, viz. "*Factorise* $(x+3y)^2 - y^2$" and "*Factorise* $x^2 + 6x + 8$". These two questions had

baulking rates of 27% and 30% respectively, and success rates of 9% and 49% respectively. As noted earlier, approximately 65% of all students who answered the first of these two questions applied the technique of *Expand and Factorise* (ie. expanding the bracketed term, collecting like terms and factorising the resultant expression). Of these students, more than one third (ie. approximately 23% of all students taking the test) failed to factorise the resultant expression; that is, their final answer was $x^2 + 6xy + 8y^2$. The reason for the low baulking rate on the difference of two squares problem is that this question can be attempted in a number of ways and, for the technique of *Expand and Factorise*, the expansion subproblem is very straightforward. For the second question, the baulking rate was marginally higher (30%), but many of those who failed to attempt this question did not attempt any of the last five questions on the test, indicating that they ran out of time. Nearly half of all students (49%) were successful in answering the question, representing a success rate of 70% for those attempting the question. By comparison, the first question had a baulking rate of 27%, but a success rate of 9% overall or 13% for those who did attempt it (see Table 4-15).

Table 4-15 Baulking and Success Rates for two Factorisation Problems

	Q5 Factorise $(x+3y)^2 - y^2$	**Q29** Factorise $x^2 + 6x + 8$
Baulking Rate	27%	30%
Overall Success Rate	9%	49%
Success Rate (for those who attempted)	13%	70%

Half of the successful students used a *Template* approach (ie. they recognised the form of the problem as a difference of two squares problem) and the other half applied *Expand and Factorise* (ie. they successfully factorised the quadratic problem). For the subgroup who applied the *Expand and Factorise* solution technique to the difference of two squares problem, one third halted prematurely, giving the answer

$x^2 + 6xy + 8y^2$. Of these students 83% successfully completed the factorisation in the (less detailed) quadratic problem, indicating that students either made a control error (after the change of goal) or are much less comfortable factorising quadratics containing two pronumerals than problems containing only one pronumeral.

Research into the nature of expertise has shown that experts tend to be stable in their problem-solving behaviours, whereas novices are more affected by contextual information and can be erratic in their performance in problem solving. This has been particularly evidenced by two questions from the diagnostic test, viz. the difference of two squares problem (question 5) and the linear equation (question 29). As the detail of a question increases, it can mask the underlying structure of the problem and inhibit recognition of the steps required for solution. For example, we observed that students were most likely to use templates for expansion problems when the coefficients of all terms took the value one, and when the power on the bracketed term was two.

In the case of the Difference of Two Squares problem, many students who had operated at the expert level by applying a template to the simple expansion problem (*Expand* $(a-b)^2$) failed to recognise the form of the new problem due to the extra detail. The added detail reduced the similarity between the given problem and the manner in which templates are typically presented to students in the classroom and in textbooks. Thus the majority of students applied the technique of *Expand and Factorise* to the problem, and many of these expanded the bracketed term by repeated application of the distributive law rather than by applying the appropriate template.

Even more dramatic was the effect of detail for the linear equation. As noted earlier, this problem is trivial if students recognise its form but it had the highest baulking rate of all questions on the test (76%) and close to the lowest success rate (9%). Again, this is related to the student's ability to perceive the basic structure of the problem, and again it has implications for the classroom in terms of the presentation of questions.

4.3.3 Confidence

The third factor affecting a student's choice of solution techniques is their level of confidence. Students with a low level of confidence show discomfort with open-ended questions (for example, Davis, 1984 or Stacey and McGregor, 1997b) and hence can be induced to apply incorrect techniques. Because they are rarely confronted with problems that cannot be solved, students expect that all questions must require *something* to be done and so can be led into making errors that they would not normally make. For example, consider question 12, viz. "*Write in simplest form* $\sqrt{a^2-b^2}$ ". This question had a baulking rate of 39% and a success rate of 11%. That is, one half of all students attempting the diagnostic test incorrectly answered this question, and of this group 86% applied a type of generalised distributivity to produce the following working $\sqrt{a^2-b^2}=\sqrt{a^2}-\sqrt{b^2}=a-b$. Only one student who scored 50% or better on the test produced this working; whereas 65% of the students who did produce this working never used generalised distributivity for any other question. That is, as a student's expertise increased (as measured by their overall performance on the test), they were less likely to be coerced into applying incorrect methods. For the remaining 35% of students who produced this working, one half also applied generalised distributivity to all of the expansion questions whilst the other half only used generalised distributivity once, viz. to expand the bracketed term in question 5 (the difference of two squares problem). This latter observation reinforces the impact that question detail can have on a student's choice of solution technique.

Another disturbing observation on this question was that 4% of students provided the answer "*c*, by Pythagoras". These students have internally represented Pythagoras' theorem in the form $a^2=b^2+c^2 \Leftrightarrow a^2-b^2=c^2$, without any understanding of the meaning of the pronumerals in the template. None of these students scored higher than 35% on the diagnostic test.

Both of the erroneous answers discussed here point to shortcomings in classroom practice and the textbooks used in secondary schools. The application of generalised distributivity to this question indicates that students need to be presented with problems that cannot be simplified, whereas the application of "Pythagoras' theorem"

indicates the importance of using graphical information whenever possible to represent algebraic ideas and of presenting templates in forms that aid their interpretation and appropriate application (ie. using symbols such as Δ rather than pronumerals). This last point has been raised as a direction for future research (see chapter eight).

4.4 Summary

In chapter two, we detailed the algebra diagnostic tests/systems and the follow-up support provided to students by several universities. We also observed that whilst such systems successfully serve the purpose for which they were developed (ie. determining the most appropriate units for enrolment), the level of diagnosis that they provide is insufficient from the point of view of student modelling within an interactive learning environment. The main problem when diagnosing errors in the domain of algebra is that errors are determined by the skills required for solving and these are, in turn, determined by the chosen solution technique. Therefore for problems that can be solved by a variety of solution techniques, a cognitive diagnostic system must be able to interpret the question and to identify all available solution techniques. The system must then be able to distinguish between the solution techniques to determine the one that was most probably adopted by the student, given that the only inputs to the system are the question and the student's answer.

For these reasons, we conducted extensive analysis of students' workings on the diagnostic test that was introduced in 1992 for students entering the Bachelor of Engineering program at the University of Ballarat. The analysis revealed that:

1. a student's choice of solution technique and their degree of success in answering a question are determined by several factors including the detail included in the question and the amount of recognition required for categorising the question, and
2. the solution technique applied to an algebra question can be determined from the *form* of the answer.

From this analysis, we developed a method of error representation that can accommodate families of errors that are not easily accommodated by a rule-based system (Sleeman, 1984). This method decomposes an answer into its individual terms, which are then classified according to their structure. Erroneous answers from 143 hand-written diagnostic tests were analysed by grouping those answers that were derived from the same solution technique. The purpose of this analysis was to determine the similarities between answers that were derived using the same solution technique and to determine the differences in structure or answers that were derived using different solution techniques.

It was found that the two most important features of a student's answer that can be used to match a given answer with those resident in the system are the **number of terms** in the answer and the **types of these terms**. Once the solution technique has been determined, other features of the student's answer such as the values of coefficients, the signs on terms, and the pronumerals and bracketed terms can be used to discriminate between variations of this technique. This analysis was undertaken to develop a model of algebra errors that would enable a computerised system to identify the solution technique that was most probably applied to a problem by examining the structure of the resultant answer. This methodology results in fine-grained diagnosis that can be explained in terms of the student's preferences for particular solution techniques and the factors that impact on these choices.

The methodology has been tested by developing a diagnostic system for algebra. In the next chapter, we detail how this analysis and our computational model of algebraic problem solving have been used to design the diagnostic system.

CHAPTER 4 STRUCTURAL ANALYSIS OF ALGEBRA ERROR DATA 117

Chapter 5 Design of the Diagnostic System

The initial project description was to develop a new approach to cognitive diagnosis and to evaluate its efficacy by designing and implementing a fully automated diagnostic system for algebra. The aim was to produce a system that did not require either multiple-choice questions or banks of hard-coded questions and answers, and that was based upon a methodology that could be applied more generally than just to this single project. Therefore, the system had to enable teachers to enter their own questions (rather than being constrained to using a database of hard-coded questions) and students to input their own one-line answers (rather than using multiple-choice questions). From these two inputs, the system would then diagnose and report the misconceptions underpinning an erroneous answer. Rather than simply reporting that a student has not mastered a particular concept or skill, it includes an identification of the individual's choice of solution technique on a particular problem and describes how the individual's problem-solving behaviour varies across an entire test. The final diagnostic system is described in chapter six.

The purpose of the current research was two-fold. Firstly, we aimed to design and implement a system that is capable of diagnosing errors made by students when solving algebra problems that are presented in standard form and are already formulated in mathematical notation. One method for implementing such a diagnostic system is to include a bank of hard-coded questions and answers. This has the advantage of having a pre-determined link between questions and answers, but the disadvantage that there is only a limited number of questions available. When questions and answers are hard-coded, only an **exact** match between two items will lead to retrieval. Several such systems are currently used to determine a student's

enrolment options at tertiary-entry level (see chapter two). However, diagnosis beyond this level is not normally realised by these systems, which brings us to the second aim of the current research, viz. to improve the level of cognitive diagnosis achieved by the diagnostic system in terms of identifying the solution protocols that an individual adopts, and the circumstances under which their problem-solving behaviour changes.

In summary, the diagnostic system must be able to identify a student's solution path and the methods used and to diagnose their errors (ie. attribute errors to either a conceptual or procedural source), to remediate the errors (using cognitive conflict) and to advise the student on other available solution techniques if they have chosen a sub-optimal method. The methodology for realising these aims is based upon the identification of a student's solution technique from their one-line answer to a problem.

The design of the algebra diagnostic system combined knowledge-level, top-down analysis (based upon the computational model) with a bottom-up, case-driven approach (based upon the error analysis). The results of the analysis of error data (chapter 4) revealed two main points. Firstly, the different solution techniques applied to a particular algebra question lead to different answer forms; that is, the *form* of a student's answer is a strong indicator of the solution technique that was applied. The most important features for discriminating between answer forms are the number of terms in the answer and the types of terms contained in the answer. Secondly, for each algebra question, there is a limited number of solution techniques that can be applied. Therefore, each question only results in a finite number of answer *forms* (although arithmetic slips can lead to infinitely many variations). These observations led to the decision to implement the diagnostic system using case-based reasoning, because this paradigm has enabled us to model the error-making process in a transparent manner. Case-based reasoning has been applied to tasks in a large array of domains and a number of researchers have emphasised the need to adopt a systematic approach to the development of such systems (for example, Bergmann and Althoff, 1998). By avoiding an *ad hoc* approach to development, the final system will be both easy to scale up (to incorporate areas of algebra not currently included) and to extend to other

domains that share some characteristics with algebra (see chapter seven for more discussion of these points).

Having decided upon an implementation paradigm, the next decision concerned how the system should be designed to enable it to determine the relationship between the question that was presented to a student and the one-line answer input by the student. The most important question to address was "How do students arrive at their choice of a solution technique?". The answer lies in the student's level of expertise (see chapter three). Research into expertise in mathematical problem solving has indicated that experts spend much time in analysing problems and that they view problems in their entirety. In turn, experts adopt optimal solution techniques such as templates. (An optimal technique is one that entails few steps and that requires little time to execute. Such techniques mean that the user can be expected to be both faster and more accurate in problem solving than users who adopt less efficient techniques.) On the other hand, novices do not take an *overall* view of a problem, but adopt means-end analysis to transform the initial state of a problem into one which is (hopefully) more amenable to solution. From these results, we found that a student's level of expertise determines the manner in which the student interprets a problem and chooses a solution technique. A classroom teacher (or the student model within an ILE) benefits from identifying how students interpret problems and choose solution techniques, because the conceptual activities undertaken during problem solving are at least as important in developing a student's mathematical thinking as the student's facility in executing procedural steps of a solution plan. These results led to the development of the computational model of mathematical problem solving, which has been used both as the basis of operation of the diagnostic system and to design the case-based reasoner.

Our system uses the structure of a student answer as the key to the diagnosis and remediation of errors. The system employs partial matching of answers when it is determining the student's solution technique, rather than exact matching. This means that arithmetic errors (eg. 11*12 = 123) do not prevent the system from matching a particular student answer with its closest counterpart in the case library. Because the

system is capable of generating new algebra questions (ie it does not employ a bank of hard-coded questions and answers), it does not have a predetermined link between a question and a particular answer. Instead, it decomposes each question (and answer) into a set of features that are then used as the indices to the relevant case base. This is done because, although there are infinitely many question variations, there is only a limited set of question types, and each question type is associated with a limited number of solution techniques.

This chapter contains the detail of the methods applied to the design of the system and the following chapter discusses how the design was implemented. The design of the system is based upon the following requirements:

- the definition of the domain,
- identification of the sources of data,
- a representation schema for the data,
- the processes that the system must undertake to generate and classify questions and to diagnose algebra errors from a single-line answer,
- the methods required for implementing the design, and
- the specific tasks and requirements of the implementation paradigm (case-based reasoning).

5.1 The Definition of the Domain

The first issue to be addressed in designing the diagnostic system was the definition of the domain. The key areas for inclusion are aspects of algebra that are normally assumed to have been mastered by tertiary-entry level students viz. *expansion*, *factorisation* and *solving linear* and *quadratic equations*. It is assumed that the user has mastery of arithmetic operations (including precedence and inverse operations), directed number, decimals and percentages, arithmetic factors and arithmetic fractions. The design of the system currently incorporates algebra questions with the keywords *Expand*, *Factorise* and *Solve*, although the last question type has not yet

been fully implemented. Questions are presented in standard form, and some limitations have been placed upon their structure. In this section, we detail the form that questions take within the system and justify the limitations placed upon these.

The domain of the diagnostic system is limited to problems which have either *Expand* or *Factorise* as the keyword. These topics are further limited - in the case of factorisation, the problem types include common factor problems, difference of two squares, difference of two cubes, sum of two cubes and general quadratics. Only problems presented in standard form are included, ie we do not have problems that require some manipulation to convert to standard form.

Expansion questions included in the system include both multiplication and exponentiation problems. Multiplication questions are those where a bracketed term is to be multiplied by a number (eg. $-2(3x - 4y)$), a pronumeral eg. $a(2b - 3a)$, or a compound term (eg. $3a^2(7ab - 12a^2c)$ or $-2x(3x - 2y)(4y - x)$). The implementation of exponentiation questions has been limited to only include problems where the power is 2 or 3 (eg. $(a - 2b)^2$ or $(2a - 3b)^3$), but can also involve multiplication (eg. $-2a(3b - 2a)^2$). For a question that comprises multiple terms to be expanded, the question is decomposed into a number of subproblems - one for each term that must be expanded.

Factorisation questions include problems from the following categories:
common factor problems (for example $2a - 6b$, $7a^2 - 12ab$, and $3(x + 4y) - 2a(x + 4y)$), grouping problems (for example, $5am - 5an + 2m - 2n$ and $x^2 + 6xy + 9y^2 - 36a^2$), general quadratics in standard form (for example, $x^2 - 81$, $x^2 + 81$ and $x^2 + 6xy + 8y^2$) and specialised cubics (sum of two cubes, eg. $8x^3 + 27$, and the difference of two cubes, eg. $x^3 - 8$).

Equation-solving questions include linear equations in a single unknown (for example, "Solve for a: $2a - 5 = 7$ " and "Solve for a: $2a - 5 = 7 + 3a$"), quadratic equations (for example "Solve for x: $x^2 - 7x + 12 = 0$" and "Solve for x: $x(x - 5) = -6$)",

exponential and logarithmic equations (for example "Solve for x: $\log_a x = b$", and "Solve for a: $e^a = b$"). Problems in non-standard form (eg. $\frac{3}{a+6} = 2a - 1$) have not yet been included in the system. Extending the scope of the system to incorporate such questions is discussed in chapter seven.

To limit memory requirements, two constraints were placed upon the construction of questions, without limiting the generalisability of the results. These are:

- the number of terms in an expression must be no more than 4, and
- the length of strings (expressions) must be no more than 64 characters.

In the next section, we provide an overview of the manner in which the system operates and detail what is required by the system to perform its tasks.

5.2 System Overview and Requirements

The initial description of this project was to develop a system that was capable of generating, classifying and storing algebra questions, receiving student answers to these questions, marking the student answers, diagnosing the errors made and producing reports that contain the full details of what the students did when solving the problems. The design of the system was based upon the model of algebraic problem solving that was developed in chapter three, and the operational phases of the diagnostic system were linked to the steps in the model. Because the system was to be completely automated, it had to be capable of using the two main inputs (the question and the student's one-line answer) to generate the diagnostic reports. This, in turn, meant that the system had to be capable of executing the steps in the model; in particular it had to be able to:

- determine what a question is asking,
- identify the set of available solution techniques,
- rank-order the solution techniques by likelihood of use,

- reconstruct the steps in the solution technique to reproduce the student's line-by-line working, and
- generate a complete diagnostic report that identified conceptual errors (which manifest as the student's choice of solution technique), procedural errors (which manifest as the incorrect execution of one or more steps in the solution plan) and control errors (which represent steps in the solution plan being executed in the incorrect order or not being executed at all).

The operation of the system was decomposed into three main functions (viz. setting a test, answering a test and printing a diagnostic report) and the decision was made to use the case-based reasoner to perform two separate tasks (viz. the classification of algebra questions and the diagnosis of student errors - these two tasks are discussed in detail in section 5.5). This overview of the system's operation is shown in Figure 5.1.

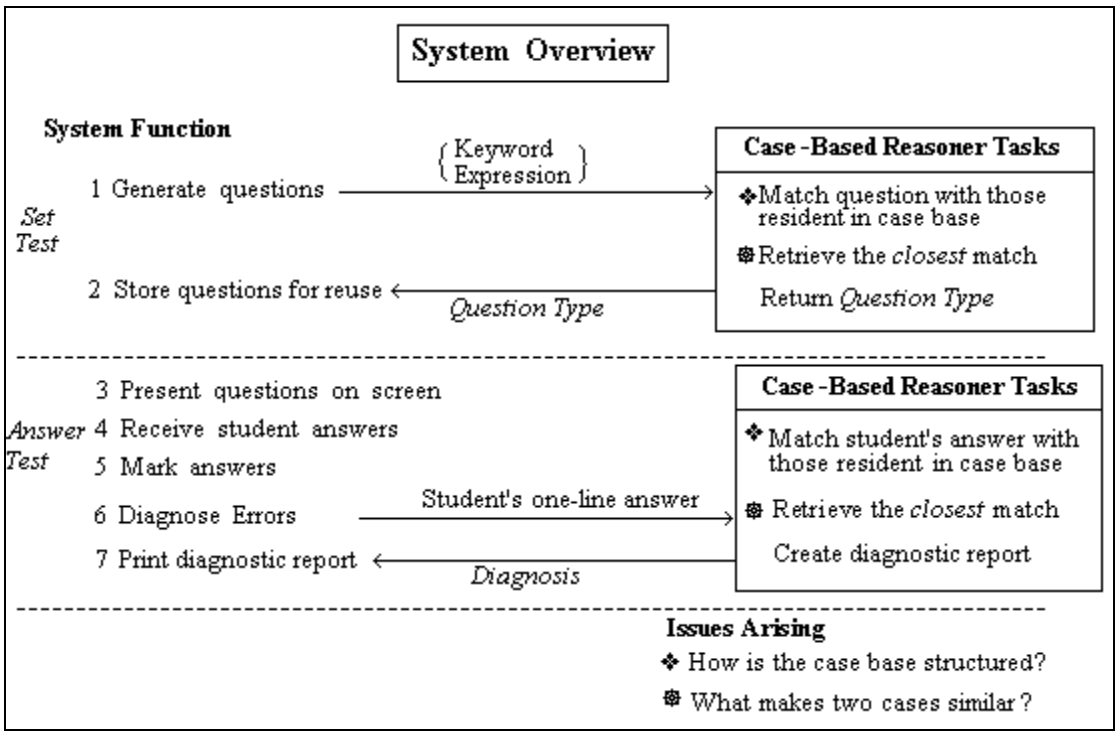

Figure 5-1 Overview of initial project description

Both the methods used for representing question types and the analysis of error data demonstrated that case-based reasoning would provide a suitable paradigm for implementation (see chapters three and four and appendix two). Several issues arose

from the initial project description and the choice of implementation paradigm. Firstly, the question arose as to how similarity matching should be performed for the two tasks (which obviously have different requirements) because the measurement of similarity is fundamental to the successful operation of any CBR system. Linked to this is the secondary issue of the choice of retrieval mechanism to ensure efficiency of operation and completeness and accuracy of retrieval. The third issue was how the case bases should be structured so that system performance was optimised in terms of both accuracy and speed of retrieval. This section provides an overview of how these issues were resolved.

Upon entry to the system, all users encounter the main menu (shown in Figure 5.2 below), which contains four items: set a test (for the teacher), answer a test (for the student), print a report (all users) and quit the system (all users). A user's status determines their rights and hence the menu options that are available to them. The three main items on the menu and their requirements are now discussed individually.

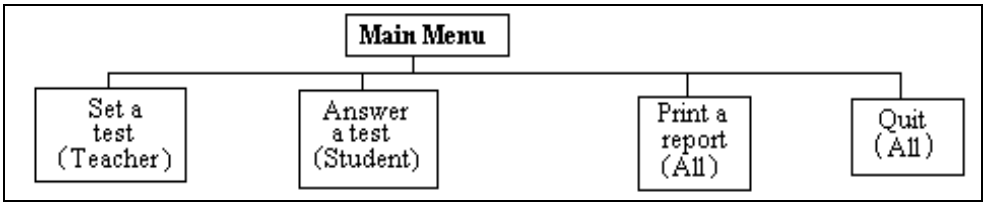

Figure 5-2 The main menu in the diagnostic system

5.2.1 Setting a Test

The menu option "*Set a Test*" is only available to teachers. It requires the teacher to enter a set of one or more algebra questions, and the system to generate the correct answers, to classify the questions and to store the data for later use. The issues to be addressed were:

- What is a question?
- How should questions be represented within the system?
- How should the CBR tasks of measuring similarity and retrieving cases be implemented?

- How should the case library be structured to maximise efficiency and accuracy of retrieval?

In this section, we outline the processes, methods and sources of data that were required to resolve these issues.

The first question was resolved by decomposing a question into the two items *Question Keyword* and *Question Expression*. The first item is chosen from a drop-down list to avoid spelling errors, whilst the second item is entered freehand by the teacher. However a single expression can be represented in a variety of formats and so to avoid redundancy in the system a single means of representation was required. This was achieved by decomposing an expression into a set of terms, ordering these by ASCII value and then recombining them to form an ordered string (see section 5.4).

From these inputs, the system must generate the outputs *Correct Answer* and *Question Type*. The first of these tasks, generating the correct answer, is achieved by using a computer algebra system (MATLAB) rather than have the teacher enter the answer. The reason for making this decision is that it is possible that the teacher could make an error when entering the correct answer, which would then impact on the marking of answers and the diagnosis of errors. Section 6.1 provides details of how this is implemented. The second task, finding the value of the attribute *Question Type*, was implemented as a classification task using case-based reasoning. The main questions to be addressed were how to implement the measurement of the similarity of questions (which is discussed in section 5.3), how to implement the retrieval of cases (which is discussed in section 5.6) and how to structure the case libraries (which is discussed in section 5.5). The inputs required for the classification phase are the keyword for the question, the ordered string corresponding to the expression to be manipulated and the feature set extracted from the expression by the parser (this set is determined by the keyword and is discussed in detail in sections 5.4 and 5.5).

The outputs from the classification phase are the unique identifier for the question (its case number) and a list of Question Types rank ordered by similarity value, where each value of the attribute *Question Type* is associated with an exemplar case file.

These data are appended to the existing question information and finally the system writes all of the test data to a text file. The design and operation of the case-based classifier are discussed in detail in section 5.4 and in the next chapter. We now detail the operation of the second menu option (viz. *Answering a Test*).

5.2.2 Answering a Test

The menu option "*Answer a Test*" is only available to students. When a user answers a test, the system must perform the following tasks: read the appropriate test file, present each question in turn and receive the user's one-line answer, allow the user to review their answers and to alter these if desired, write the user's answers to a text file, mark each question as either correct or incorrect, generate a test mark, identify those questions for which the user requires diagnosis, to perform the diagnosis and to write the diagnostic report to a file. The issues to be addressed were:

- What is an answer?
- How should answers be represented within the system?
- How can the marking of answers be implemented?
- How should the CBR tasks of measuring similarity and retrieving cases be implemented?
- How should the case library be structured to maximise efficiency and accuracy of retrieval?

In this section, we outline the processes, methods and sources of data that were required to resolve these issues.

The first two questions impact on the third and were resolved by decomposing an answer string into a set of terms, ordering these by ASCII value and then recombining them to form an ordered string (see section 5.4). Marking answers in an algebra system is usually performed in one of two ways: using a computer algebra system or evaluating the answer string over a grid of points. Neither of these two methods was considered to be suitable for the current system because they both fail to distinguish between strings such as $(x+y)(x-y)$ and x^2-y^2 . However this distinction is fundamental to the success of the current system. To overcome this problem marking of answers is

effected by using string-matching functions and the ordered strings for both the correct answer and the student's answer (as for question expressions). This method is satisfactory for comparing strings that contain terms with integer coefficients but not for comparing strings that contain terms with non-integer coefficients. In the latter case, comparison of answer strings can be performed by decomposing each term in an answer string into its components (viz. the term sign, coefficient, pronumeral list and bracketed term list) and comparing these components.

The last two questions refer to the implementation of the diagnostic phase as a CBR task (which is discussed in detail in section 5.4 and in the next chapter). The main questions to be addressed were how to implement the measurement of the similarity of answers (which is discussed in section 5.3), how to implement the retrieval of cases (which is discussed in section 5.6) and how to structure the case libraries (which is discussed in section 5.5). The inputs required for the diagnostic phase comprise the outputs from the classification phase (ie. the question keyword and the ordered expression string, the ordered hypothesis list of *Question Types* and the case number for the question under investigation), the student's answer string and the feature set extracted from it by the parser (this set is determined by the *Question Type* and is discussed in detail in sections 5.4 and 5.5).

The outputs from the diagnostic phase is a list of possible diagnoses rank ordered by similarity value. Data contained in a diagnosis include an explanation of how the student interpreted the question, the identification of the *Solution Technique* that was most probably adopted by the student and the corresponding line-by-line working that led to the student's answer. These data are appended to the existing information contained in the student's answer file and finally the system writes the complete test data to a text file that is used to generate the report files. The design and operation of the case-based diagnoser are discussed in detail in section 5.4 and in the next chapter. The operation of the third menu option (viz. *Printing a Report*) is now outlined.

5.2.3 Printing a Report

Reports are stored as text files, and which reports a user can access is determined by their user rights. Students can only view and print their own reports, whereas teachers can access both the individual student reports plus the compiled reports generated for an entire class. When the user opts to print a report, it is first written to the screen and is printed when the user responds to the system prompt. The form that each report takes is similar to those provided by *DIAGNOSYS* (see chapter two).

Representing mathematical data in a meaningful manner is fundamental to the success of the diagnostic system. Issues that needed to be addressed include determining what makes two questions similar, what makes two answers similar, and how the system can convert both input types into a form that can be utilised within the system. In the next two sections, we discuss how the system evaluates the similarity of questions and answers including a discussion of the work performed by the parser.

5.3 Similarity Matching

A successful CBR system is one that retrieves the cases that are most like the current problem and that does this as quickly as possible. The level of any system's success is determined by the algorithms used for measuring the similarity of cases and retrieving relevant cases from its case library. The case-based component of the diagnostic system must be able to identify data elements (either questions or answers) that are identical and elements that are similar but are not identical. In this section, we detail how the system measures the similarity between two data elements.

5.3.1 Similarity of Questions

The diagnostic system must be able to identify two questions that are identical (so that if a question is being reused the system can reuse the associated data) and, for non-identical questions, determine which questions are most alike. Measuring the

similarity of questions requires that questions have a standard method of representation.

So, what is an algebra question? As described earlier, a question can be represented by a pair of attributes: the keyword and the expression to be manipulated (for *Solve* questions a third attribute, the pronumeral of interest, is also required). For two questions to be **identical**, they must have the same keyword and the two expressions must be *mathematically equivalent*, if not identical. This last point raises the issue that we cannot use string matching alone to perform equivalence testing on the expressions. For example, the two expressions in Figure 5-3 are mathematically equivalent but not identical. Generating the unique expression string is one of the tasks performed by the parser. At this point, it is sufficient to note that for two expressions to be mathematically equivalent, they must contain the same sets of terms, where a term comprises a number of subterms, a sign, a coefficient, a list of pronumerals and their powers and a list of bracketed terms and their powers.

$$\boxed{\begin{array}{l} a^2 - 2ab + b^2 \\ a^2 + b^2 - 2ba \end{array}}$$

Figure 5-3 Two strings that are mathematically equivalent but not identical

Assessing the similarity of two questions is a much more complex task than determining equivalence. In chapter three, we described the model of mathematical problem solving upon which the system is based and discussed the implementation of the four stages (*Classify*, *Plan*, *Execute* and *Review*). The first of these stages involves the classification of questions and we showed that questions with different keywords require different representations. That is, the keyword is the feature that has the greatest discriminatory power (see Figure 5-4).

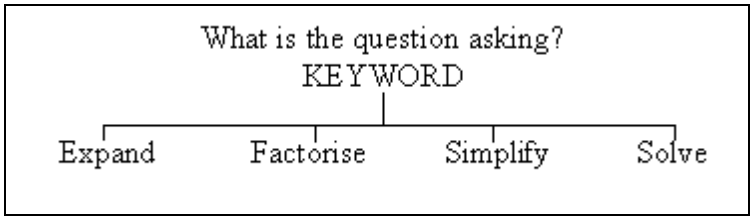

Figure 5-4 Importance of the keyword in determining the similarity of two questions

Assume now that three questions share a single keyword, but that no two expressions are identical. Determining the similarity of the expressions requires the system to identify the structure of the expressions and the operations implied by the structure. For expansion questions, the implicit operation (multiplication or exponentiation) determines both the set of solution techniques available and the skills required to perform each of these. At the next level, other features (such as the actual values of coefficients, pronumerals and bracketed terms) are required for determining the applicability of different solution techniques. By comparison, factorisation questions require the identification of a factor common to all terms, the number of terms in the expression, and the type of each term (eg. numerical, linear, quadratic etc.). These features require the system to also identify lower-level features (such as the actual values of coefficients, pronumerals and bracketed terms). The weight given to each of these features is determined by their discriminatory power and is discussed in more detail in section 5.5 and in chapter six.

5.3.2 Similarity of Answers

Matching answers within the system takes two forms: **exact** matching (for marking tests) and **partial** matching on the form of answers (for retrieval in the case-based reasoner). The first type of matching is achieved by string comparison of the ordered strings. However partial matching of answers requires the system to decompose an answer string into a set of terms, which are then classified. The number of terms and the list of term types are then used as the indices to access the *Solution Technique* case base. A list of term types recognised by the system can be found in Table 5-1.

Table 5-1 Definition of Term Types

Term Type	Number of pronumerals	Number of bracketed terms	Powers on pronumerals	Powers on bracketed terms	Example
linear_n	0	0			-5.2
linear_p	1	0	[1]		-2*a
linear_b	0	1		[1]	3*(2a - b)
linear_cross_b	0	2		[1, 1]	(2-y)*(4y+3)
linear_cross_p	2	0	[1, 1]		9*a*b
linear_cross_bp	1	1	[1]	[1]	7*a*(3a+ 4b)
quadratic_b	0	1		[2]	3*(2a-b)^(2)
quadratic_b_cross_p	0	1		[2]	3*v*(2a-b)^(2)
quadratic_p_cross_b	1	0	[2]		3*v^(2)*(2a-b)
quadratic_p	1	0	[2]		-2*a^(2)
cubic_b	0	1		[3]	3*(2a-b)^(3)
cubic_b_cross_p	0	1		[3]	3*a*(2a-b)^(3)
cubic_p	1	0	[3]		-2*a^(3)
cubic_p_cross_b	1	0	[3]		3*a^(3)*(2a-b)

In this section, we have described the features that are used to measure the similarity between two questions and those that are used for matching answers. In the next section, we detail how different data sets are represented within the system.

5.4 Representation of Data in the Diagnostic System

As demonstrated in the previous section, the system must be capable of recognising expressions that are mathematically equivalent. It must also be capable of extracting the relevant feature sets required for matching and classifying questions and answers.

To realise these requirements, we developed the parser, which serves two main purposes. Firstly, when questions and answers are represented as strings, the mathematical equivalence of expressions is lost. To overcome this, the parser converts each string into a standard form, so that the mathematical equivalence of two strings can be tested using string comparison functions. Secondly, the parser converts each question and answer into a form that can be matched with cases stored in the case libraries (the operation of the parser is discussed in chapter six).

It is also important that all data sets generated by the system are stored in a form that can be accessed by different computer systems and other software packages (for example, a teacher may wish to perform analysis of data collected from an entire group). For this reason, text files have been used to store the data. In the remainder of this section, we outline how algebra questions and answers have been represented within the system.

5.4.1 Representing Algebra Questions

The parser is used to convert each question into the required format, which includes the keyword, the original expression string, the ordered expression string (required for matching within the case base), the number of terms and a list for each decomposed term. The term lists contain eight attributes: the ordered term string, the number of subterms within the term, the term sign, the numerical coefficient (default is 1), the number of pronumerals, a list containing each pronumeral and its power, the number of bracketed terms, and a list containing each bracketed term and its power. An example is shown in Figure 5-5.

Input Factorise $a^2 + b^2 - 2ba$

Final Structure

Keyword	Factorise
Input string	$a^2 + b^2 - 2ba$
Ordered String	+a^(+2)+b^(+2)-2*a*b
Number of Terms	3
Term 1	+ a^(+2) , 1, +, 1, 1, [a, +2], 0, [ ]
Term 2	+ b^(+2) , 1, +, 1, 1, [b, +2], 0, [ ]
Term 3	-2*a*b , 3, -, 2, 2, [a, +1, b, +1], 0, [ ]

Each term is decomposed into a list:

[Ordered String , Number of Subterms , Term Sign , Coefficient , Number of Pronumerals , Pronumeral List , Number of Bracketed Terms , Bracketed Term List]

Figure 5-5 Structure of an algebra question

5.4.2 Representing Answers

Similarly, the parser is used to convert each answer (both correct answers and student answers) into the required format by extracting each term and generating both the ordered answer string and a list of term types. The ordered strings are required for **exact** matching, which is used when the system marks a test and determines which answers are incorrect and require diagnosis. On the other hand, the case-based component uses the *structure* of an answer for the purpose of retrieving the solution plan that was most probably adopted by the student. In chapter four, we demonstrated that this structure can be represented by two features: the number of terms in the answer and the set of term types. An example of answer representation is shown in Figure 5-6.

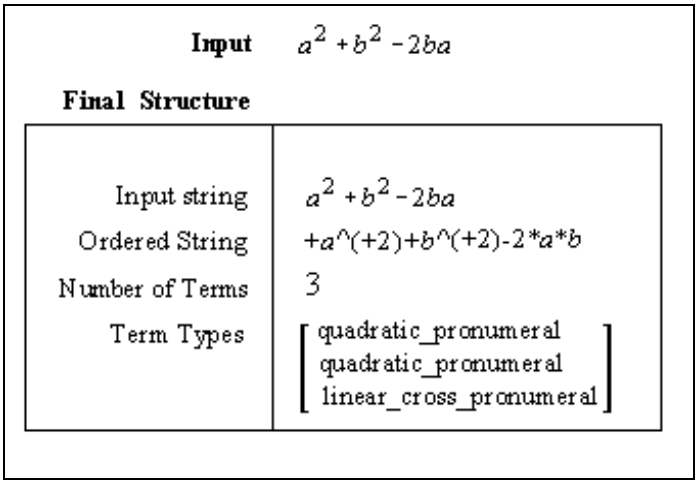

Figure 5-6 Structure of an answer

Having determined the best manner to represent the data within the system, we next considered how to structure the case libraries. In the next section, we detail how the two case libraries were designed (details of the implementation and operation can be found in chapter six).

5.5 Design of the Case-Based Reasoner

In section 5.2, we discussed some issues arising from the initial project description concerning how the case bases should be structured, which cases should be stored and how to evaluate the similarity of both questions and answers. For any CBR system, a case is a set of specific knowledge tied to a context. The fundamental problem involved in CBR is how to *index* cases to enable efficient and appropriate retrieval. To represent cases, we need to identify what to store, what structure to use for description, how to organise and index the case memory and how to integrate the case memory structure into the model of general domain knowledge. We also require effective methods for searching and matching in the case base. The indices for cases within the library are crucial in determining the success of retrieval, because a case's indices identify the circumstances under which it is appropriate to retrieve the given case. That is, the problems of case representation and indexing are interrelated and

should be addressed simultaneously. In this section, we detail how the cases in the CBR have been designed (details of implementation can be found in chapter six).

The case-based reasoner was designed to implement the different stages of the computational model of mathematical problem solving (see chapter three). Operation of the system has two major phases that employ the case-based reasoning component, viz. test entry and error diagnosis. Because the CBR component has two main functions, it is structured in two layers (viz. the *Question Types* and the *Solution Techniques* case libraries), each of which has a different basic case representation (see Figure 5-7). The test-entry phase involves the categorisation of questions, whilst the diagnostic phase involves the determination of the solution plan that was most probably adopted by the student.

The first component of the case-based diagnostic system implements the *Interpret Problem* component of the computational model. It is a library of cases each of which represents a single question type (for example, Difference of Two Squares) and the set of features associated with the given question type (including the problem goal and a set of features such as the number of terms, term types, number of bracketed terms and powers applied to bracketed terms). Cases retrieved from the *Question Type* case base represent the possible interpretations of a question that the CBR hypothesises the student may have made. All matches above a threshold value are placed on a hypothesis list and combined with the set of features extracted from the initial expression. These are then passed to the *Solution Technique* case-based planner, commencing with the highest matched case.

Operation of Case-Based Reasoner

CBR Task	Input	Operation	Issues
Classify Questions	Question	Group related (similar) questions Find correct group to which the current question belongs Search the correct group and find the *closest* match Retrieve *Question Type*	What makes two questions similar ?
Diagnose Errors	Student's one-line answer	Match student's answer with those resident in case base Find the *closest* match Create Diagnostic report	What makes two answers similar ?

Figure 5-7 Overview of the issues involved in designing the case-based reasoner

The case-based planning system implements the *Plan Solution* and *Execute Plan* phases. It represents all known solution techniques associated with a particular question type. Each case in this repository includes a plan as a sequence of steps that represents a worked solution to the problem. These generative mechanisms are stored as a part of each case within the *Solution Techniques* case-base, and are used to validate the match between a student's answer and one stored in the library. For a single-step problem, the case represents an individual skill contained in the solution plans, and is associated with a set of procedural errors (or mal-rules).

In the remainder of this section, we detail the structure of the case-based reasoner and how cases are represented and indexed in its two components.

5.5.1 The Question Type Case Base

In any case-based system, the components of a *stored* case include a description of the problem, a description of the solution and a description of the outcome achieved by applying the solution. The *Question Type* case base is a set of three files (one for each keyword). The components of a *stored* case include a description of the problem and a

value for the attribute *Question Type*. For the *Question Type* case base, a *query* case is simply a description of the new problem, represented by the keyword and the set of problem features extracted by the parser.

Once the diagnostic system has extracted these features, they are used as indices to access the *Question Type* case base. The CBR system compares the description of the current question with those for the cases currently residing in the case base and retrieves the closest matching cases. The *Question Type* cases are generalised cases; thus they do not need to include specific details relevant to a particular question and different question types require different sets of features for identification. Each question type is associated with an exemplar case, which comprises an algebra question and the set of answers that have been given by past students. It is this case that is used during the diagnostic phase (see section 5.4.2).

When a question is generated, it is represented as a set of three or four strings: the keyword of the problem, the pronumeral of interest (only required for *Solve* questions), the expression to be manipulated and the correct answer. The expression is parsed to extract the set of features required for accessing the cases in the *Question Type* case base. The retrieval mechanism accesses the stored cases by nearest neighbour retrieval on the feature set. However retrieval cannot be a rigid application of nearest neighbour because the weights of the features are not pre-determined but are dependent upon the current user's level of expertise. A data structure called a discrimination network represents the relationship between features in a way that enables their relative weights to be determined. Different discrimination networks are used for each problem goal (keyword). This is reflected by the structure of the *Question Type* case base, which is divided at the top level into separate libraries for each keyword (see Figure 5-4). This was done because question types are determined by both the keyword and a set of surface features extracted from the problem, but the choice of which features to extract is determined by the problem goal.

In the remainder of this section, we detail how the different libraries are structured and provide an overview of how these are used by the case-based reasoner (see chapter six for full details).

5.5.1.1 Representing Expansion Questions

The discrimination network used to represent expansion questions is shown in Figure 5-8. For expansion problems, the feature with greatest discriminatory power is the operation to be applied to the bracketed term (either multiplication or exponentiation). For this reason, the question "*What is the operation*?" is at the top of the discrimination network (and the feature *Operation* is given the highest weight - see chapter six). Depending upon the answer to this question, we need to consider different features at the next stage. For multiplication problems, the next most important feature is the *Type of the Multiplier*, which can be another bracketed term, a pronumeral, a number or a composite type; whereas for exponentiation problems, the value of the *Power* has greatest discriminatory power.

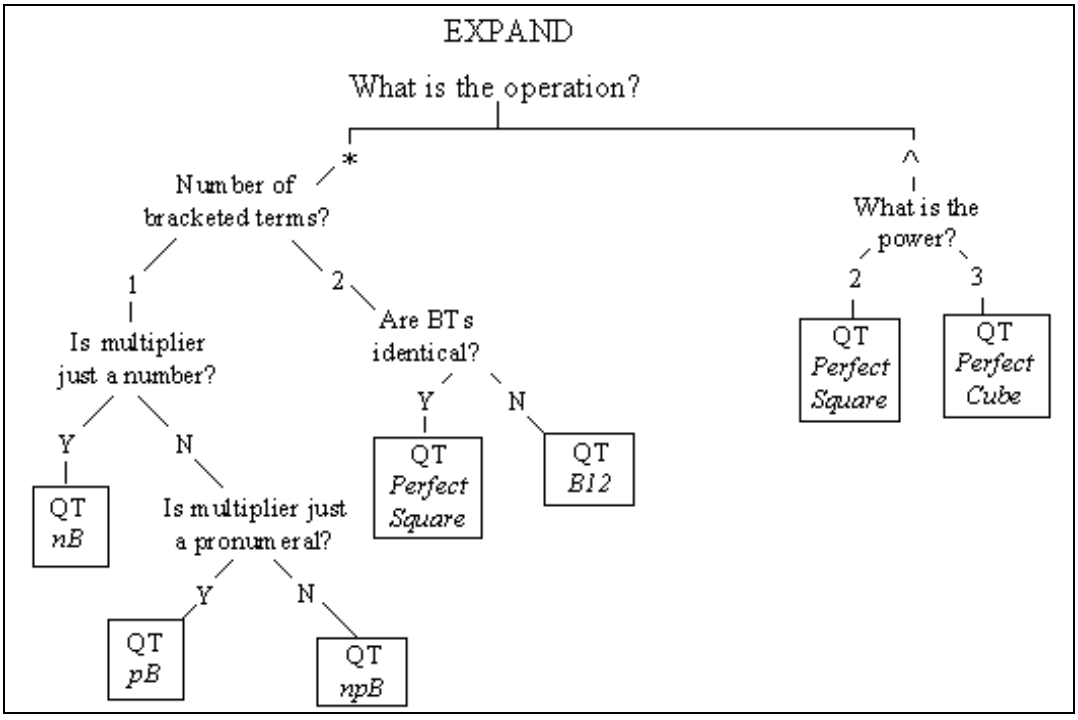

Figure 5-8 Discrimination network for Expansion problems in the *Question Type* case base

From Figure 5-8, we derived a number of *Question Types* for expansion problems. Those for expansion problems involving multiplication are described in Table 5-2. For expansion problems involving exponentiation, there are only two question types incorporated in the diagnostic system. These are *Perfect Squares* and *Perfect Cubes* (ie. we have not included powers greater than 3, although very little work is required to extend the system to include these).

Table 5-2 Question Types with keyword "Expand" and operation is multiplication.

Question Type	Description	Examples	Solution Techniques
EnBm	number*Bracketed Term	$2(3a+4b)$	DLn
EpBm	pronumeral*Bracketed Term	$c(3a+4b)$	DLp
EnpBm	number*pronumeral*Bracketed Term	$2c(3a+4b)$	DLnp
EPSm	Perfect Square $\Delta.\Delta$ where Δ is a bracketed term	$(a+b)(a+b)$	• TPS • FOIL • RDL
EB12m	$\Delta.\Theta$ where Δ and Θ are different bracketed terms.	$(a+b)(c+d)$	• RDL • FOIL

Linked to each *Question Type* is a set of available of *Solution Techniques.* Those associated with multiplication expansion problem types are detailed in Table 5-3 and those for exponentiation expansion problems are contained in appendix two.

Table 5-3 Solution Techniques pertaining to expansion problems involving multiplication

Solution Technique	Description	Examples	Steps
DLn	Apply distributive law with numerical multiplier	$2(3a+4b)$ $=6a+8b$	1. Multiply n by the coefficient of each term inside the bracketed term.
DLp	Apply distributive law with pronumeral multiplier.	$a(d-2c)$ $=ad-2ac$ $a(3b-a)$	1. Concatenate *a to end of each term inside the bracketed term. 2. Sort strings. 3. Apply first index law by finding repeated pronumerals and summing powers. 4. Rewrite and sort strings.
DLnp	Apply distributive law for numerical multiplier followed by that for pronumeral multiplier where pronumeral is not part of bracketed term.	$2a(3a-4b)$ $=a(6a-8b)$ $=6aa-8ab$ $=6a^2-8ab$	1. DLn 2. DLp

RDL	Repeated application of the Distributive Law after separating terms in first bracketed term.	$(2x+3y)(a-b)$ $=2x(a-b)+3y(a-b)$ $=2ax-2bx+3ay-3by$	1. Separate terms inside first bracketed term. 2. Multiply each of these terms by the second bracketed term by applying one or more of DLn, DLp, and DLnp. 3. Collect like terms. 4. Sort terms.
TPS	Template for expanding a perfect square: $(\Delta+\Theta)^2=\Delta^2+2.\Delta.\Theta+\Theta^2$	$(2x+3y)(2x+3y)$ $=(2x)^2+(3y)^2+2(2x)(3y)$ $=4x^2+9y^2+12xy$	1. Identify Δ and Θ. 2. Calculate and by applying index laws. 3. Calculate 2(Δ)(Θ) by nB and pB. 4. Sort terms.
FOIL	First Outside Inside Last	$(a+b)(c+d)$ $=a\times c+a\times d+b\times c+b\times d$ $=ac+ad+bc+bd$	1. Multiply first terms in each set of brackets. 2. Multiply outside terms in each set of brackets. 3. Multiply inside terms in each set of brackets. 4. Multiply last terms in each set of brackets. 5. If required, apply index laws to each resultant term. 6. Collect like terms. 7. Sort strings.

Similar analysis was conducted for algebra questions with the keyword “Factorise”. This analysis is now detailed.

5.5.1.2 Representing Factorisation Questions

Factorisation problems are more complex to analyse than are most expansion questions, because most expansion problems have very few associated solution techniques and these are self-evident. That is, for a problem such as *Expand* $3ax(4x-9a)$, the student does not need to engage in a search for a solution technique. Compare this with a problem such as *Factorise* $16ab^3-54a(g+2h)^3$, which requires several iterations to effect full factorisation. To focus on the system's performance at diagnosing errors, we made several assumptions/decisions. These are: 1) limit of four terms, 2) general cubics are not included but two special cases (Difference of Two Cubes and Sum of Two Cubes) are, and 3) we have restricted the system to only consider real-valued functions (eg. the expression x^2+81 has complex factors but no real factors so the system must accept both NULL and $(x+9i)(x-9i)$ as correct answers for the problem *Factorise* x^2+81).

In chapter three, we outlined a four-step, rule-based approach to solving factorisation problems that is included in many secondary school text-books. This methodology can be represented by the redundant discrimination network shown in Figure 3-2. However, we wanted to represent data in a manner that can be more efficiently accommodated within our case-based classification system. For this reason, we have represented the classification of factorisation problems by the discrimination network shown in Figure 5-9. This network has two main advantages over that shown in Figure 3-2: firstly, it contains no redundancy and secondly, we can represent different levels of expertise by simply traversing the tree to different depths (the higher the level of expertise, the deeper the search of the network).

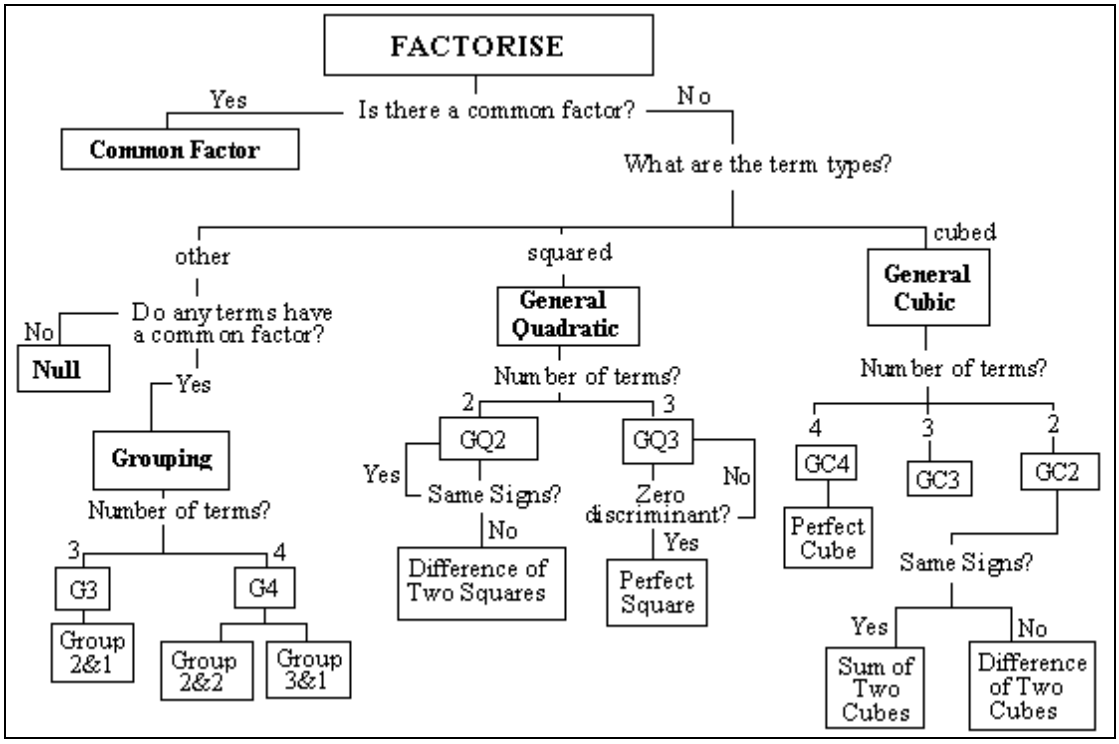

Figure 5-9 Discrimination network for representing factorisation problems

The most important feature of *any* factorisation problem is the existence of a factor common to all terms. Hence the first question in the discrimination network is "*Is there a common factor*?". The system must mimic this performance. That is, for any expression to be factorised, the system must first identify whether a common factor exists. To achieve this, the system first identifies the greatest common divisor of the coefficients. Pronumeral common factors and bracketed term common factors are then identified. However, students do not always recognise and remove a common factor from an algebraic expression. Consider the problem *Factorise* $2x^2 - 8y^2$ and the two sets of working shown in Figure 5-10. Both students recognised the form of the problem as being a Difference of Two Squares problem and correctly applied the template, although only student A recognised the presence of the common factor at the outset.

Student A	Student B
$2x^2 - 8y^2$ $= 2(x^2 - 4y^2)$ $= 2(x-2y)(x+2y)$	$2x^2 - 8y^2$ $= (\sqrt{2}x - \sqrt{8}y)(\sqrt{2}x + \sqrt{8}y)$ $= \sqrt{2}(x-2y) * \sqrt{2}(x+2y)$ $= 2(x-2y)(x+2y)$

Figure 5-10 Two sets of working for a factorisation problem

For this reason, the existence of a common factor does not limit a factorisation problem to being classified *only* as a common factor problem. Instead, such questions are flagged as containing a common factor but are then classified by structure to determine if it also falls into another category. Table 5-4 contains details of the types of factorisation problems that are incorporated into the system. The related Solution Techniques are detailed in appendix two.

Table 5-4 Question Types with keyword "Factorise".

Question Type	Description	Examples	Solution Techniques
CF	Common Factor	$6a + 9b$ $-6ab - ac$ $7a^2b - 28ab^2$ $4p^2qr - 2pq^2r^3 + 6pqr$	RCF
D2S	Difference of Two Squares $\Delta^2 - n$ $\Delta^2 - \Theta^2$	$x^2 - y^2$ $x^2 - 7$ $(x+2y)^2 - (2x+3y)^2$ $(x+2y)^2 - (2m-5p)^2$	• TD2S • GDL

D2C	Difference of Two Cubes $\Delta^3 - n$ $\Delta^3 - \Theta^3$	$x^3 - y^3$ $x^3 - 27$ $(x+2y)^3 - (2x+3y)^3$ $(x+2y)^3 - (2m-5p)^3$	• TD2C • GDL
S2C	Sum of Two Cubes $\Delta^3 + n$ $\Delta^3 + \Theta^3$	$x^3 + y^3$ $x^3 + 27$ $(x+2y)^3 + (2x+3y)^3$ $(x+2y)^3 + (2m-5p)^3$	• TS2C • GDL
GQ	General Quadratic	$6x^2 + 2x + 3$ $3x^2 - 7$ $2x^2 + x$ $2(x+3y)^2 + 3(x+3y) + 1$ $e^{2t} + 5e^t + 4$ $m^2 + 81$	• HQ • QF • GDL • E&F • NULL
Grouping	Group terms to find common factors	$ax + 5y - 5a - xy$ $m^2 + 2m + 1 - p^2$ $(p+1)^2 - p - 1$	• G21 • G22 • G31

Of the standard factorisation problems outlined in Table 5-4, *General Quadratics* and *Grouping4* problems require further discrimination before we can determine suitable *Solution Techniques*. The features for discriminating between these are detailed in Figures 5-11 and 5-12.

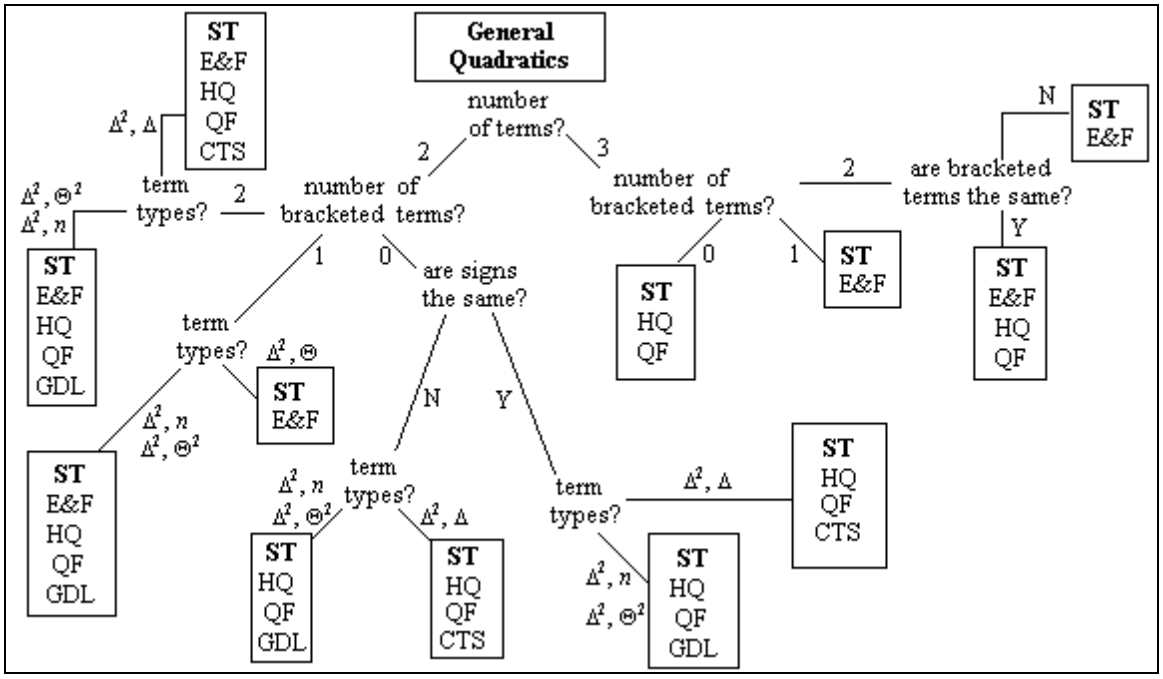

Figure 5-11 Discrimination network for Solution Techniques for factorising General Quadratics

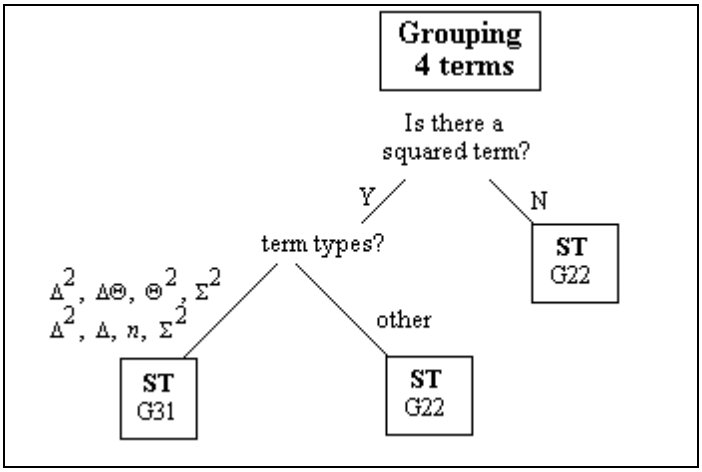

Figure 5-12 Discrimination network for Solution Techniques for factorising Grouping problems

Once the system has extracted the feature set from a particular question, it must determine the specific question type for the problem. Because we are interested in how a student categorises a question, we want to retrieve all "reasonable" matches for *Question Type*, not just the most specific case. For the factorisation problem *Factorise* $(x+3y)^2 - y^2$, typical classifications include NULL (the student believes that the question cannot be factorised), General Quadratic and Difference of Two Squares

These are shown in Figure 5.13, along with the surface features of the question that influence students in generating these categorisations.

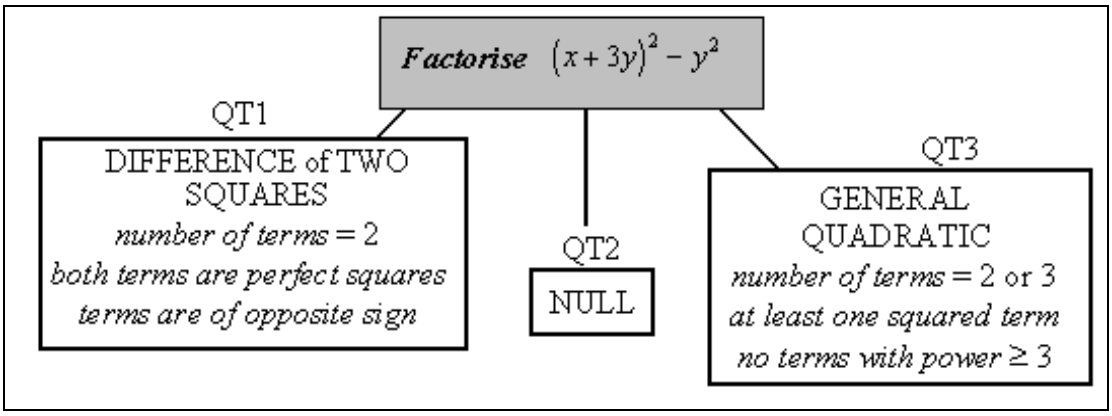

Figure 5-13 Typical student classifications of the question *Factorise* $(x+3y)^2 - y^2$

Similar analysis was performed on all the types of factorisation questions included in the system, and the results were used to determine the *Question Type* case representations (those for General Quadratic and Difference of Two Squares are shown in Figures 5.14 and 5.15). Although General Quadratic expressions can take a variety of forms, only the one relevant to our example is included here. However, the form does affect the choice of solution technique and so both the *Question Type* and the *Solution Technique* case bases contain separate cases for each variation of a general quadratic. If a student categorises a particular question as some other type, the system will need to learn these new values and, if appropriate, add them to its case base. Determining which cases should be stored requires input from the domain expert. This issue is discussed in more detail in the next chapter.

Attribute	Value
Question Type	FD2Spb (Difference of Two Squares with a bracketed term and a pronumeral term)
Keyword	Factorise
Number of terms	2
Signs on terms	+ -
Term Types	quadratic_bracket, quadratic_pronumeral
Greatest common divisor of coefficients	1
Number of pronumerals common to all terms	0
Number of bracketed terms common to all terms	0
Exemplar Case	155

Figure 5-14 Case frame for *Question Type* = Difference of Two Squares with bracketed term and pronumeral term

Attribute	Value
Question Type	FGQ2pb (General Quadratic with two terms - a bracketed term and a pronumeral term)
Keyword	Factorise
Number of terms	2
Term types	quadratic_bracket, quadratic_pronumeral
Signs on terms	NA
Greatest common divisor of coefficients	1
Number of pronumerals common to all terms	0
Number of bracketed terms common to all terms	0
Exemplar Case Number	216

Figure 5-15 Case frame for *Question Type* = General Quadratic with bracketed term and pronumeral term

Associated with each Question Type is an exemplar problem; the case number for the associated exemplar is one attribute recorded in each of the case frames in the *Question Type* library. Although there are only a very limited number of problem

types, there are infinitely many variations for each. To improve the efficiency of the diagnostic system, we decided to include a single exemplar that can be used to represent a particular *Question Type*. This decision and its impact on the system structure are now explained in more detail, along with the design of the *Solution Technique* case base.

5.5.2 The Solution Technique Case Base

The diagnostic stage involves the identification of the solution technique adopted by a student from all the available solution techniques, which include typical erroneous methods as well as valid ones. To do this, the system combines the retrieved values for the *Question Type* (starting with the most highly ranked value on the hypothesis list), the associated exemplar case, the set of surface features extracted from the question and a set of features extracted from the student's answer. The system then retrieves the exemplar case associated with the question type.

Each answer in the exemplar case is represented both as an ordered string and as a set of features, which is used to determine which of the associated answers most closely matches that given by the student. Further, each answer in the exemplar case is linked to a solution plan, which is a *generalised* case comprising the steps required to obtain the particular answer. A sample exemplar file is shown in Figure 5-16.

case_number 84
keyword Factorise
expression +(+3*y+x)^(+2)-y^(+2)
correct_answer +(2*y+x)*(+4*y+x)

Ordered Answer	**Number of terms**	**Term Types**	**Generative Mechanism**
+(+2*y+x)^(+2)	1	quadratic_b	c84-1.kb
+(+3*y+x-y)^(+2)	1	quadratic_b	c84-2.kb
+(+3+x)^(+2)	1	quadratic_b	c84-3.kb
+(+3*y+x+y)*(+3*y+x-y)	1	linear_cross_b	c84-4.kb
+(+3*y+x+y)+(+3*y+x-y)	2	linear_b; linear_b	c84-5.kb
+(+2*y+x)+(+4*y+x)	2	linear_b; linear_b	c84-6.kb
+6*x*y+8*y^(+2)+x^(+2)	3	linear_cross_p; quadratic_p; quadratic_p	c84-7.kb

+6*x*y+9*y^(+2)+x^(+2)-y^(+2)	4	linear_cross_p; quadratic_p; quadratic_p; quadratic_p	c84-8.kb
+3*x*y+x^(+2)	2	linear_cross_p; quadratic_p	c84-9.kb
+(+3*x*y)^(+2)+x^(+2)-y^(+2)	3	quadratic_b; quadratic_p; quadratic_p	c84-10.kb
+5*y^(+2)+x^(+2)	2	quadratic_p; quadratic_p	c84-11.kb
+8*y^(+2)+x^(+2)	2	quadratic_p; quadratic_p	c84-12.kb
+2*y^(+2)+x^(+2)	2	quadratic_p; quadratic_p	c84-13.kb
+9+x^(+2)	2	linear_n; quadratic_p	c84-14.kb
+3+x^(+2)	2	linear_n; quadratic_p	c84-15.kb
+y^(+2)*(2*y+x)^(+2)	1	quadratic_b_cross _p	c84-16.kb
+(3*y-y)*y+(+6*y+x)*x	2	linear_cross_bp; linear_cross_bp	c84-17.kb

Figure 5-16 A sample exemplar case file

This plan (or generative mechanism) is the key to identifying both the skills required for solving the question and the individual errors that emerge when the student executes their solution plan. Each step in the solution plan corresponds to a single skill that needs to be performed. The adaptation phase instantiates the plan in terms of the query question. The plan is then executed and the final answer is parsed. If there is an exact match between the student's ordered answer string and that from the exemplar case, we can reuse the diagnosis that was generated for the exemplar case. If not, other plans associated with the exemplar case are tested. If no match is found for the given *Question Type* and exemplar case, then other cases from the hypothesis list of *Question Types* are tested. After each step of the solution plan has been executed, the state of the problem is updated to reflect the hand-written working that the student may have generated whilst solving the problem. When a match is made between the student's answer and that generated by the system, the path that the student followed when solving the problem can be reconstructed and is used to form the basis for the

reports. If there is no match between the student's answer and any answers stored in the exemplar files that were examined, the system must accommodate the new case.

For the factorisation example in Figure 5-13, the correct *Question Type* is FD2Spb (ie. this is a Difference of Two squares problem with a pronumeral term and a bracketed term) and its associated exemplar file is case 84, which has a number of typical solution plans to be examined. These techniques include the Difference of Two Squares Template, Generalised Distributivity, Expand and Factorise, Quadratic Formula and Quadratic Heuristics. Consider the example where the student correctly applied the technique *Expand and Factorise*, but failed to factorise the resultant expression., leading to the final answer $16a^2 + 48ab + 11b^2$. The working generated by the solution technique *Expand and Factorise* is shown in Figure 5-17.

Attribute	**Value**
Keyword	Factorise
Expression	$4(2a + 3b)^2 - 25b^2$
CorrectAnswer	$(4a + 11b)(4a + b)$
QType	FD2Spb
Exemplar case	84
ThisAnswer	$16a^2 + 48ab + 11b^2$
Solution Technique	Expand and Factorise
Step 1	Expand Bracket
Output 1	$16a^2 + 44ab + 4ab + 36b^2$
Step 2	Collect Like Terms
Output 2	$16a^2 + 48ab + 11b^2$
Step 3	Factorise
Output 3	$16a^2 + 48ab + 11b^2$
Errors	Failed to factorise.

Figure 5-17 Working produced by the generative mechanism for Expand and Factorise

To validate the proposed diagnosis, the system presents the hypothesised working to the student. If the student agrees with the diagnosis, it is written to the student's answer file. Otherwise, the system needs to accommodate the new case in its case library. This is discussed in greater detail in section 5.7.

The next section discusses the issue of case retrieval, and outlines how it has been addressed within our system.

5.6 Case Retrieval

For ill-defined problems, it is common to use a statistical "nearest neighbour" algorithm to match and rank stored cases with a query case. In this approach, the system uses a vector of features to describe cases, where each feature is assigned a weight that reflects how important it is in matching cases. Each feature of the input case is then compared with the corresponding feature in the stored case and a similarity score for that feature is calculated. The type of a feature (string, integer, Boolean etc.) determines how the comparisons are made. For example, the value of a Boolean variable in the new case either matches that for a stored case (and is assigned a similarity score of 1) or does not (and is assigned a similarity score of 0). A weighted aggregate score is then calculated for all of the stored cases and these scores are used to rank the stored cases.

The main problem with this approach is that retrieval cannot commence until a similarity score is calculated for *every* case within the library. Also, the set of retrieved cases can vary significantly if changes are made to the weights assigned to the features. For well-defined problems, the most common approach is inductive retrieval, which involves the clustering of related cases to improve the efficiency of retrieval. By clustering cases, retrieval is made more efficient because only a subset of the case library needs to be compared with the new problem case. Given a set of classified cases, induction provides a means of deriving the rules that led to the classification.

The diagnostic system uses a hybrid approach to case retrieval. Testing of our initial prototype indicated that, for our domain, inductive retrieval is superior to nearest neighbour retrieval. Performance of the inductive retrieval process was improved by

creating our own qualitative model of the domain knowledge, which was used to group related cases and to guide index generation. A qualitative model is a means of developing summaries of features as well as defining causal relationships between case features. This model led us to cluster cases such as those with the same keyword in the *Question Type* case base, and related answers in the exemplar case. When the system has determined which case library to explore, it then applies standard nearest neighbour retrieval (see the next chapter for detail of the implementation). The improvement in retrieval performance that emerged from the use of the qualitative domain model was expected because of the highly-structured approach to mathematical problem solving and the hierarchical nature of the domain knowledge.

In the next section, we outline how maintenance and extension of the case-based component of the diagnostic system have been implemented.

5.7 Maintenance and Extension of the System

A case-based reasoner "learns" by indexing new cases and adding them to its memory. Cases worth remembering are those that differ in some way from past cases. The system also needs to store some normative knowledge and some means of determining whether the differences are significant enough to make the new case worth storing. Because our system has two case libraries with different purposes and structures, the maintenance processes also vary. Case reuse focuses on differences between past and present cases and what *part* of the retrieved case can be transferred to the new one. That is, it is important that the system be able to identify whether an error has resulted from incorrect question categorisation, inappropriate choice of solution technique or simply an incorrect execution of an appropriate technique. We now outline what maintenance of the different case libraries entails.

5.7.1 Maintenance of the Question Type Case Base

Algebra questions can be categorised very tightly. That is, we do not anticipate that the *Question Type* case base should need to be updated to accommodate new categorisations for the set of problems currently included in our system. However, in future, we would like to extend the capability of the system so that it can incorporate problems that are not in standard form, for example a problem such as *Solve* $\frac{3}{(x+1)} = x + 2$ for x (English, 1997).

5.7.2 Maintenance of the Solution Technique Case Base

Adaptation of the *Solution Technique* case library is potentially very difficult to achieve because of the great variety of errors that students make (Davis, 1984). Students, particularly early learners, can be very creative in producing their own solution techniques, so it is important that we identify those new methods that are worth storing. The most important step is to determine whether an error made by a student represents a misconception (in which case it should be stored) or a random slip (in which case it should not be stored). Repair theory states that students may produce new solution methods when they reach an impasse during problem solving (see Brown and Van Lehn, 1980). However, unless a method is repeated, it is more likely to represent a patch than a genuine new technique. This is the criterion used to determine whether or not to store a new case. Similarly, we expect that future students will generate new procedural errors, ie. errors in executing a single step of a standard solution plan. Again, repetition of the error on a complete test will be the best indicator of whether or not the case library needs to be expanded to incorporate the new case.

Maintenance involves generating complete explanations for all errors present in a student's response based on causal knowledge of errors. If a new *procedural* error is encountered, it is incorporated in the existing exemplar case as a generative mechanism along with an explanation and suggested remediation, both of which are stored in the associated text file. However, the system also needs to be able to accommodate new *conceptual* errors. These are much more difficult to incorporate

into the system, because we need to be able to engage the student in a dialogue to trace their reasoning at each stage of the problem-solving process (Moore, 1995). In particular, the system needs to be able to determine how the student classified the question and chose their solution technique. These are items that the student may not be able to articulate in the form required by the system, so an alternative method (viz. having the student enter a line-by-line solution) has been employed. The solution method that led to the answer must be inferred from this working. This stage requires that the system works with the domain expert (usually the teacher) to validate the new case and add it to the case library. To safeguard the integrity of the system, maintenance is not automated, rather it always involves the domain expert.

5.7.3 Extension of the System

Because it is modular in design, the system should be easily extended to incorporate extra modules with only some modifications to the parsing phase. This may require some duplication of content to improve searching efficiency. In particular, the design of the *Solve* case library in the *Question Type* will be implemented (see appendix two for details of the analysis that has already been conducted). The next step will be the inclusion of problems presented in non-standard form, such as equation-solving problems that are presented as algebraic fractions. This will require us to extend the parsing phase as well as the *Question Type* and *Solution Technique* case libraries.

5.8 Summary

The diagnostic system has been designed using the computational model of mathematical problem solving developed in chapter three and the results of the error analysis conducted on students' workings on algebra problems (see chapter four). This chapter presented the details of the methods and processes required to realise the aim of implementing a diagnostic system that can identify and explain the errors made when answering algebra questions. Methods were required to implement the non-CBR tasks involved in setting and answering tests. These tasks included entering questions,

obtaining the correct answer for a question, storing question data, reading the question data from storage, presenting questions on screen, receiving student answers, allowing students to review their answers, marking answers and storing a student's question-answer data. The two tasks that employed the case-based reasoners were the classification of questions and the diagnosis of errors. These tasks required methods for representing queries, measuring the similarity of two cases, retrieving appropriate cases, adapting the cases and maintaining the case libraries.

System operation is divided into three main elements: test entry, answering a test and printing diagnostic reports. The first operation involves the teacher in generating a set of questions. Correct answers are provided by a computer algebra system to avoid errors that would arise if the teacher supplied the incorrect answer. The case-based reasoner performs its first task (classifying questions) at this point. Although a question may be used any number of times, it only needs to be classified once, so this task is performed at test entry, and the output from the classification phase is appended to the text file where the test questions are stored.

The second stage of operation is answering a test. After a student submits their answers to the system, they are marked. All questions that were incorrectly answered are sent to the second component of the case-based reasoner for diagnosis. Other inputs to this stage are provided by the output from the classification phase. Associated with each question type is an exemplar case, which contains details of all previous student attempts at answering the particular type of question. It is the exemplar case file that is used for diagnosis. The student's answer is compared with all of the answers contained in the exemplar case, searching for the closest match by using two structural features: the number of terms and the types of the terms. For answers that have the same structure, the system uses the output from the associated generative mechanisms to determine the solution technique that was probably adopted by the student. When a match is made, the diagnostic report associated with that answer is used to provide the basis for the diagnostic reports to students and teachers. If no match is made, the system must acquire the new case; this step always involves a domain expert to protect the integrity of the case libraries.

The next chapter contains details of how the system design has been implemented.

Chapter 7 System Evaluation

Whilst expert systems have been increasing in number and application over the last twenty years, the development of formal evaluation methodologies for such systems has not kept pace. This situation has been addressed over the last five years or so; for example, questions of validating knowledge-based systems in the area of medical diagnosis were explored in December 1995, when over 100 researchers from computing, medicine, and medical informatics convened at the National Library of Medicine. The group noted that the issues surrounding the topic of system evaluation are very broad and dependent upon the purpose of the particular system under investigation (Menzies 1997a, 1997b).

One issue that is common to the developers of all expert systems is that of **when** to evaluate. Evaluation of the algebra diagnostic system has been conducted at all stages of its development, and consequently the system design has evolved throughout the development process. By gradually developing system components, we have been able to identify problems as they arose and through experimentation and continual change have been able to identify and meet the needs of the system and its users. The current system is intended to be used by students and their teachers at both upper secondary and tertiary levels. However, the methodology used to develop the system should also have application beyond its current state of development. The evaluation of the system has focused on the following issues:

- determining how the system can improve problems experienced by learners of algebra and their teachers,
- system design and its strengths and weaknesses,
- ability of the system to scale up,

- ability of the methodology to be applied to areas other than those included in the current system, and
- the performance of the system in diagnosing algebra errors (in terms of both its consistency and accuracy).

In this chapter, the evaluation of the system is detailed. This begins with a discussion of the formative evaluation that was conducted in terms of the evolution of the system and includes a description of the various prototypes that were developed. This is followed by a discussion of the needs of a computerised system if it is to be capable of achieving effective cognitive diagnosis. These needs, and the manner in which they have been addressed in the current research, form the basis of the remainder of the chapter, which includes an assessment of the design and performance of the system when diagnosing algebra errors.

7.1 Formative Evaluation and Evolution of the System

Although this thesis is presented in a linear manner, the design and development of the system were conducted simultaneously with the development of the model of mathematical problem solving and the analysis of errors on the hand-written diagnostic tests. Chapter four provided details of the error analysis conducted on the hand-written working provided by students on the Diagnostic Test conducted at the University of Ballarat in the period 1996-1999. The analysis indicated that a student's solution technique could be inferred from the *structure* of the answer. The model of mathematical problem solving that was presented in chapter three, and is summarised in Figure 7-1, identified the question and the student's answer as the two main inputs to the diagnostic system. However an interim step is required in the process, viz. the classification of the problem under investigation, which identifies the *Question Type* (the third input to the diagnostic process). The operation of the final system follows the stages shown in Figure 7-1. In this section, the evolution of all stages of the research are discussed in terms of the system prototypes that were developed.

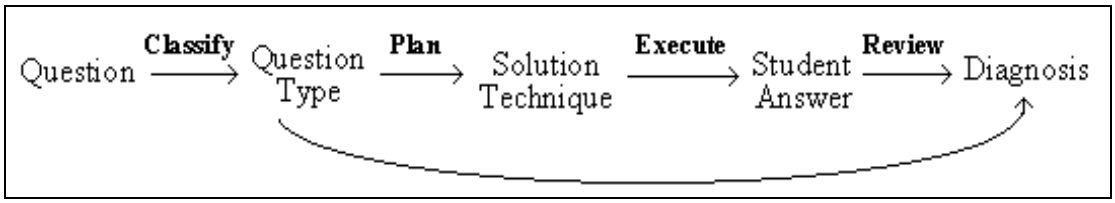

Figure 7-1 Model of mathematical problem solving

The system evolved from a set of hard-coded question-answer pairs into its final form, which is fully automated. To achieve this, a series of experiments (or implementations) was conducted. The experiments were aimed at addressing a number of focus questions:

- What is a case?
- What features are required to represent a case?
- What weights should be assigned to each of the features?
- How should similarity be measured?
- How should retrieval be implemented?
- What methods, functions and processes are required for automation?

This section details the answers to these questions and the evolution of the system.

7.1.1 Prototype A

After the initial error analysis was conducted, the first prototype of the diagnostic system was implemented using ReMind®[1], a proprietary case-based reasoning shell. The decision to use an existing CBR shell for this task was taken so that the focus was on the issues involved in case representation and retrieval. Our decision to use ReMind was based on its availability. The prototype contained a single library that included all question-answer pairs that were obtained from the ten algebra questions from the initial (1996) Diagnostic Test (see chapter four). Each case was hard-coded and included eight features or fields (see Table 7-1 and Figure 7-2).

[1] ReMind® was a product of Cognitive Systems Incorporated.

Table 7-1 Fields in the cases in Prototype A

Attribute	Type	Description
Question	Text	Single entity comprising the keyword and expression entered by the teacher.
Question Type	Text	Entered by hand.
Student Answer	Text	Single string entered by student.
Abstracted Student Answer	Text	Single answer string in postfix notation.
Correct?	Boolean	Indicated whether a particular answer was correct.
Full Answer?	Boolean	This field was used to distinguish incomplete (but correct) answers, eg. returning a single answer to a quadratic equation that has two correct roots.
Solution Technique	Symbol	Chosen from list.
Explanation	Text	Entered by hand. This field was used to form the basis of the diagnosis.

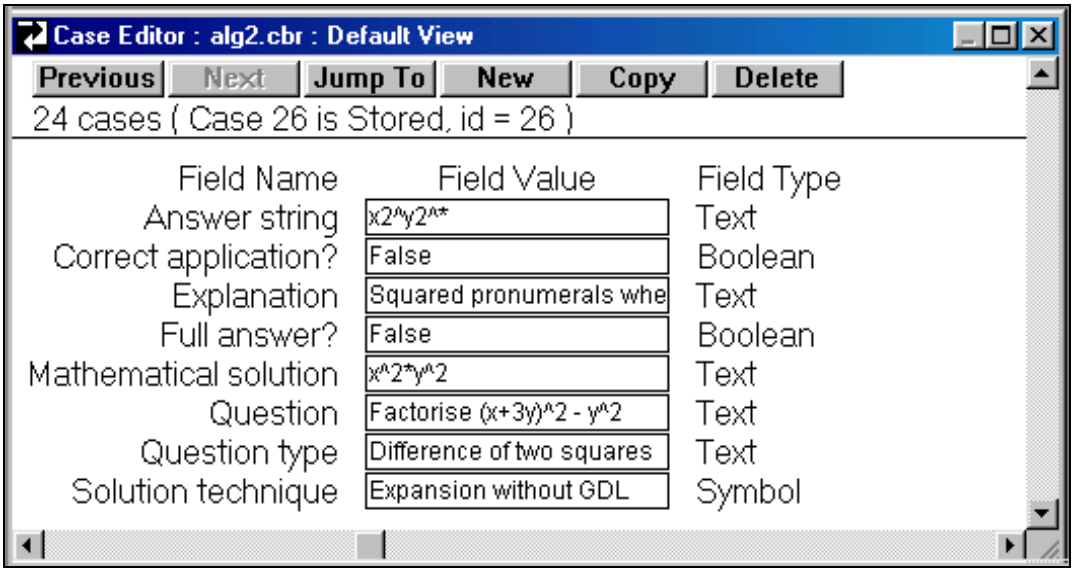

Figure 7-2 A case from Prototype A

Three of the eight fields (*Question*, *Question Type* and *Abstracted Student Answer*) were selected as the fields for matching. Other features included *Student Answer* (the answer string as entered by the student), *Correct*? (a Boolean to indicate whether or not the student's answer was correct), *Full Answer*? (a Boolean to indicate whether or not the student's answer was complete), *Solution Technique* (a description of the solution plan) and *Explanation* (which included the steps in the solution plan and the errors made). Because of the flat library structure, retrieval employed the Nearest Neighbours algorithm.

Because most of the fields were of type *Text*, string matching was used. Due to the impoverished representation, the system had poor discrimination between cases except those that matched **exactly** on the three fields. Testing was limited by several factors and the results identified the need to include some details of the **structure** of the question to enable the system to determine *Question Type* (because the structure of a student's answer varies with the *Question Type*), the need for a better library structure to exploit the hierarchical nature of the domain, the need for different case representations for different question types (because different features are required for identifying different question types), the need for a better means of matching cases than simple string matching and the related need for a single means of representing mathematically equivalent expressions other than postfix notation. This last point eventually led to the construction of the question parser (see section 6.2), while the need for improved answer representation led to the construction of the second prototype, which is discussed next.

7.1.2 Prototype B

The second prototype of the diagnostic system was also implemented using ReMind®. To address the problems identified by testing the first prototype, we began by identifying the key features to include in each case. For diagnosing errors, the most important feature of an algebra problem is the solution technique employed by the student because it determines the skills required to solve the problem and hence also

determines the types of errors that the student may make. This feature can then be used to guide retrieval of relevant cases from the case library and hence was common to all cases. Similarly, all cases included fields for the student's input (as a text string), the processed answer string (see section 6.2), a Boolean field indicating whether or not the student's answer was correct and a text field to explain the student's misconceptions and associated errors. This last feature was included because one of the strengths of the final system is its ability not just to identify errors but to explain them. This requires the identification of the solution technique used, the skills which have not been mastered and the particular misconceptions involved. The explanation for a current case can then be generated by adapting explanations for related cases and the output can then be used to target remediation to the individual user.

The new prototype comprised separate libraries for each of the ten algebra questions to enable the use of different representations for the answers to each question. Because of the flat library structures, retrieval again employed the Nearest Neighbours algorithm for the relevant library. The focus of the experiment was the matching of answers as a means of identifying the student's *Solution Technique*. Each case was again hard-coded with a different set of features for each *Question Type* (although seven of the original features were common to all questions; these were *Question*, *Question Type*, *Student Answer*, *Abstracted Student Answer*, *Correct*?, *Solution Technique* and *Explanation*). The *Abstracted Student Answer* was obtained by converting the *Student Answer* to a postfix format.

Other features, which were determined by the structure of the students' answers and were peculiar to the question under investigation, were also used. For example, the field *Sum of powers=3*? was included in all cases for expanding the cubic, and two fields *Correct CrossProduct*? and *Correct y-squared term*? were used when investigating the case where the student adopted the Expand and Factorise Solution Technique on the problem of factorising the difference of two squares (see Figure 7-3). Some simple problems required no extra features, for example the problems

"Solve $e^a = 3b$ for a" and "Solve $\frac{3}{x} - \frac{4}{a} = \frac{5}{b}$ for x" required only the seven basic features.

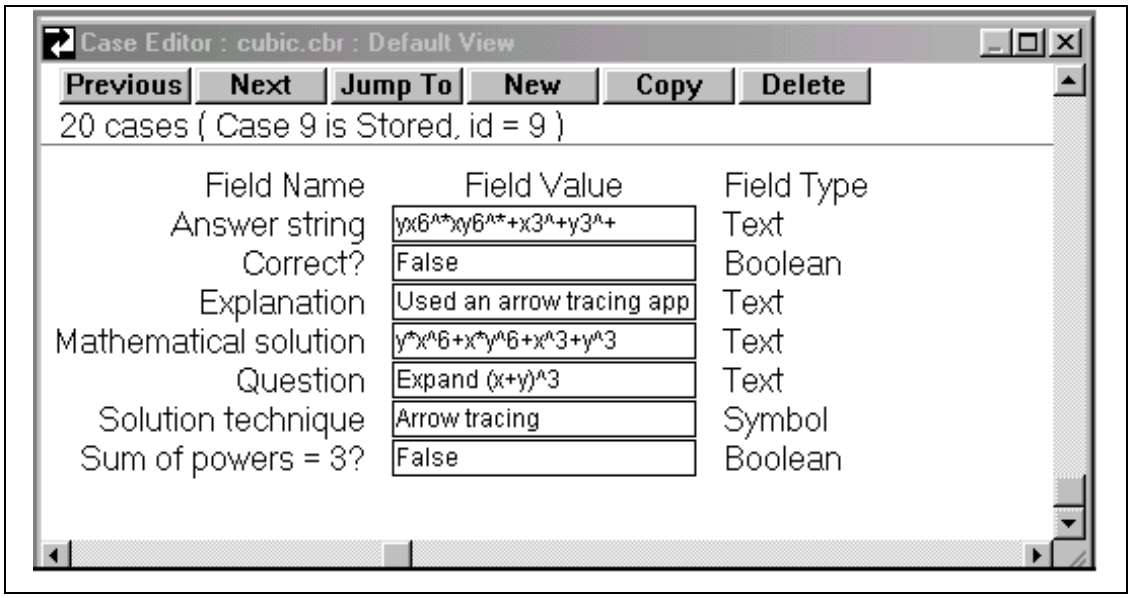

Figure 7-3 A case with the Expand and Factorise Solution Technique in Prototype B

The results of this experiment were similar to those from the first experiment, ie. the system was only useful for **exact** matching on both *Question* and *Abstracted Student Answer*, and discrimination between other cases was very poor. This again highlighted the need to include some details of the **structure** of the question to enable the system to automatically determine the value of the attribute *Question Type* for a particular problem and to improve the efficiency of retrieval. The first step in implementing this requirement was to splinter the question into two components, viz. keyword and expression. This change was implemented in the third prototype which is discussed next.

7.1.3 Prototype C

The third prototype was again implemented using ReMind® and with a separate library for each *Question Type* (based upon the keyword). For the first time, the system included questions other than those from the initial Diagnostic Test. This experiment had as its focus the identification of the features and methods that would

enable the system to automatically determine *Question Type*. Although all cases were again hard-coded, this experiment also aimed to improve library structure by using clustering of cases rather than simply using a flat structure.

The features used to represent the problem and the student's answer were determined by the value of *Keyword*. For example, the case frame for Factorisation problems is shown in Table 7-2 and a sample frame is shown in Figure 7-4.

Table 7-2 The fields in a case from Prototype C

Attribute	**Type**	**Description**
Question Expression	Text	The expression string as entered by teacher
Question Type	Text	Entered by hand
Student Answer	Text	Single string entered by student
Abstracted Student Answer	Text	Single string in postfix notation
Correct?	Boolean	
Full Answer?	Boolean	
Solution Technique	Symbol	Chosen from list
Explanation	Text	Entered by hand this field was used to form the basis of the diagnosis
Discriminant (required by all quadratics)	Symbol	Value not required - simply whether it is greater than, less than or equal to zero
Number of terms	Integer	
Term Types	Symbol	Chosen from list
Signs on terms	Symbol	Chosen from list
Coefficients on terms	Real	Used for common factor problems

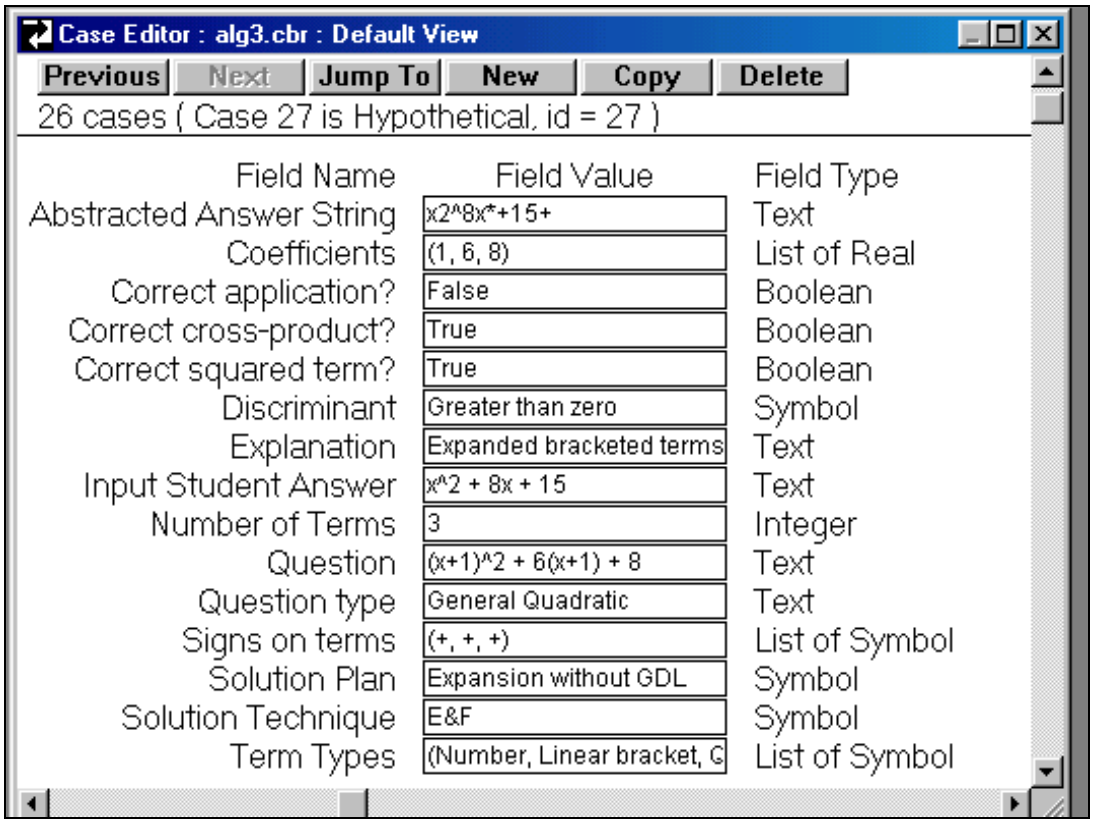

Figure 7-4 A case frame from Prototype C

In testing the prototype, both Nearest Neighbours and Inductive retrieval techniques were used. In Nearest Neighbour matching in the ReMind shell, the user creates a vector of importance values (or weights) for all the features of a case. Each feature of the input case is then compared with the corresponding feature in the stored case and a similarity score for that feature is calculated. The method for calculating the similarity score is dependent upon the data type of that feature. An overall measure of similarity between the two cases is then calculated as the weighted average of the feature similarity scores. The major drawback with this method is that retrieval cannot commence until a similarity score is calculated for every case within the library, which can become inefficient for very large libraries.

Inductive retrieval involves the clustering of related cases to improve the efficiency of retrieval. Given a set of classified cases, induction provides a means of deriving the rules which led to the classification. ReMind provides the user with the facility to

build their own qualitative models of the domain knowledge to guide index generation. A qualitative model is a means of developing summaries of features as well as defining causal relationships between case features. If a qualitative model is not defined, ReMind builds its own cluster tree using the CART algorithm (Breiman *et al*, 1984).

Our aim was to choose the most appropriate retrieval mechanism and to refine case representation so that the system would locate and retrieve all relevant cases. This was only possible because we had complete knowledge of the contents of the case base. Inductive retrieval was conducted using both the cluster tree built by ReMind and our domain model, whereas a number of trials were conducted using nearest-neighbour retrieval and different sets of weights for the fields. For each trial, we found that inductive retrieval was most accurate when we used our domain model (ie. it only retrieved those cases that we judged to be the most relevant). We also found that Nearest Neighbour retrieval always gave highest ranking to the same set of cases that inductive retrieval judged to be most on-point. Whilst changing the weights given to case features affected the similarity scores for each case, it made little difference to the rankings of the cases with highest similarity scores. We also found that while a flat library structure was satisfactory for the initial prototype, as the size of the system grew, performance in searching and retrieval degraded. Performance of the inductive retrieval process was improved by creating our own qualitative models of the domain. This was expected because of the hierarchical nature of the domain knowledge and supported our decision to exploit this feature in the revised system. However, it was also found that further decomposition of questions was required as many features were not relevant to all questions.

At this point, the decision was taken to separate the tasks of problem classification and error diagnosis. The first stage was to implement the classification task using another paradigm (neural networks) to test whether the feature set chosen to represent factorisation problems had sufficient discriminatory power to classify the problems correctly.

7.1.4 Experiment with Neural Networks

Because the classification task was to be separated from the task of diagnosis, an experiment using neural networks to classify factorisation problems was conducted (Mays and Day, 1999). One of the major strengths of neural networks is pattern recognition (or data classification), where the system is provided with a vector of features (or inputs) and then maps these features onto the class or category that they most closely match. For this reason, it was decided to explore the ability and limitations of neural networks in matching factorisation problems with standard problem forms. The aim was two-fold: firstly, to determine whether a neural network could outperform the case-based system in classifying problems using the same set of problem features (if not, the feature set used to represent algebra problems would need to be expanded), and secondly, whether the network could simultaneously determine the available solution techniques. It was expected that while a neural network should be capable of performing pattern matching, its ability to categorise mathematical problems could be limited by the lack of mathematical knowledge captured in the feature set. It emerged that the network performed remarkably well - in fact better than expected, given its limited domain knowledge - but that its performance could be improved by using a richer feature set.

A sample of 325 factorisation questions was extracted from a number of secondary-level textbooks. The questions were divided into two sets of problems (one each for training and testing the network). These included questions of the following types: simple common factor problems (numerical and algebraic), perfect squares, difference of two squares, general quadratics, grouping problems and null problems (ie. problems which cannot be factorised over the real numbers). Solution techniques (other than null) included: removing a common factor, applying a template (for example, the template for factorising the difference of two cubes), applying the quadratic formula, applying heuristics, using an approach based on generalised distributivity, and finally expanding part of the question and factorising the resultant expression. The sets of questions were chosen so that (nearly) equal number of each problem type were included in the two sets.

The inputs to the system were:

- the list of signs on the terms,
- term types encoded as the powers on the pronumerals in each term,
- the number of bracketed terms,
- the list of powers on the bracketed terms, and
- a Boolean variable indicating whether the bracketed terms were the same (to enable the system to identify perfect squares).

The neural network was set up as a fully-connected, feed-forward network with a topology of 59-10-10. The 59 input nodes covered all the chosen problem features while the 10 output nodes represented the question type and the available solution techniques (see Tables 7-3 and 7-4). The hidden layer contained 10 nodes.

Table 7-3 Coding *Question Types* in the neural network experiment

Question Type	Representation
Common Factor	0000
Grouping	0001
General Quadratic	0010
Perfect Square	0011
Difference of Two Squares	0101
Difference of Two Cubes	0110
Sum of Two Cubes	0111
NULL	1000

Table 7-4 Coding *Solution Techniques* in the neural network experiment

Solution Technique	Representation
Template	000
Expand and Factorise	001
Generalised Distributivity	010
Heuristics	011
Quadratic Formula	100
Remove Common Factor	101

The expression to be factorised in each question needed to be represented by a unique input pattern. That is, each feature that we had identified as being necessary for categorising questions had to be encoded in a binary format. To restrict the size of the input pattern and to minimise errors in encoding the algebraic expressions, we placed several limitations on the form of the expressions without loss of generality. These were:

- Each expression was limited to a maximum of four terms. The sign on each term was either N (null term), + or -. A null term indicated that there were fewer than the maximum number of terms for a given expression, eg. an expression such as $4x+6$ has 2 positive terms and so the sign list was represented as ++NN.
- Each expression was limited to a maximum of four different pronumerals.
- Powers on pronumerals were integers no greater than 5, where a power of zero indicated that a particular pronumeral did not occur as part of a given term. Therefore the actual pronumeral did not need to be included, only its power within each term.
- Each expression was limited to a maximum of two bracketed terms.
- Powers on bracketed terms were integers no greater than 2.

These restrictions enabled us to limit the number of input nodes to 59 as follows:

- signs on terms required 6 nodes because there was a total of 30 possible patterns,
- term types, which comprised the powers on the pronumerals in each term, required 48 nodes,
- the number of bracketed terms required 2 nodes,
- powers on bracketed terms required 2 nodes, and
- 1 node was needed to represent the Boolean variable indicating whether bracketed terms were the same.

For example, the algebraic expression $ab + 4ab^2 + 6a^2b$ contains 3 terms (all of these have a positive sign and the "fourth" term has a null sign), two pronumerals and no bracketed terms. We decided to represent the signs in a question with three positive terms by the six-node pattern 0,0,1,1,1,1. The powers on *a* in the terms are 1, 1, 2 and 0, whilst for *b* they are 1, 2, 1 and 0. Each term was broken down into a 12-node pattern, which represented the powers of four pronumerals, for example the first term was reduced to the pattern 0,0,1,0,0,1,0,0,0,0,0,0. Therefore, this expression was uniquely represented by the total input pattern:
[00111100100100000000101000000000100100000000100100000000000].

To train the network, a sample data set of factorisation problems was taken which included examples of the different question types. For each problem in the training set, the correct question type and the complete set of available solution techniques were encoded. Although the network trained fairly well, it did not perform well when distinguishing between problems that required mathematical knowledge; for example, it could not distinguish between general quadratics and perfect squares, nor could it identify numerical common factor problems. This problem caused a lack of performance when distinguishing some patterns, so a slight bias was made in the data set which comprised 64 problems. Each problem was represented in a binary format that the neural network computer program (SNNS[2]) could read. The network was trained to various levels of error rate so that it could be tested with a set of test data to locate its optimal performance.

To test the network, a test data set of 46 examples was created with several different problems of each question type. This set of problems was chosen to be as similar as possible to the training set in terms of the numbers of problems of each type. The data was tested at various levels of error rate. Testing revealed that the network performed optimally at a 5% error rate (this was the lowest error rate that it could be trained to given the impoverished data representation). The testing also showed that the network could be used to determine different solution techniques that were valid for a particular problem. This finding means that the network could possibly be used for the task of diagnosis when the system is unable to determine how a student obtained a particular incorrect answer.

In summary, the network could not classify numerical Common Factor problems (because it lacked information about term coefficients, typically these problems were classified as Grouping problems), Perfect Squares problems, Grouping problems, General Cubics or NULL problems (it seemed to default to a classification of Common Factor). However, it accurately classified algebraic Common Factor and General Quadratic problems. Its success rate was 85%. In this regard, it did not outperform the case-based prototypes. The network's performance in identifying

[2] SNNS is a shareware neural network package that runs on UNIX systems.

available solution techniques was less impressive, with a success rate of 74%. In particular, the network failed to identify the correct solution techniques for problems that it had classified as being of type Grouping, Perfect Square or Difference of Two Squares. For the latter two instances, the network correctly identified a subset of the available solution techniques, whereas it consistently returned inappropriate techniques for Grouping problems. However, it is true that students often fail to identify Grouping problems and so the techniques that the network identified could provide a useful basis for diagnosis when a case-based system fails in the diagnostic phase.

The results of the experiment indicated that, regardless of the implementation paradigm, the feature set used to represent problems should be supplemented with some extra detail (for example, the values of coefficients and the actual pronumerals occurring in each term). At this stage, several decisions were made. Firstly, the decomposition of questions had to be improved in order to enable the Question Type to be found automatically. This meant that the system would require methods for identifying, counting and classifying terms. Secondly, once terms were identified, they had to be decomposed to identify the value of the sign, the coefficient, all pronumeral terms and all bracketed terms. Thirdly, the system needed to be able to determine the range of available Solution Techniques for individual problems which required a means of classifying problems in terms of the expression structure, not simply the keyword. Finally, the flow of data within the system had to be determined so that the link between a student's choice of solution technique and the structure of the final answer could be exploited by the system. The choice was made to use pure case-based reasoning as the implementation paradigm.

These decisions led to the development of the model of mathematical problem solving (see chapter three) and the parser (see chapter six). Once these were developed, the decision was made to separate the tasks of classification and diagnosis. Each question could be answered by many students but need only be classified once, therefore classification was to take place at the time of setting a test and the results of classification would be stored as part of the test. The value of the attribute *Question*

Type would thereby be available as an input to the diagnostic phase. This led to the final representation scheme and the development of one final prototype before the final system was implemented; these implementations are now detailed.

7.1.5 Prototype D

The final prototype was implemented using CBR-Works[3] (a proprietary CBR shell that allows for an object-oriented approach when modelling the domain) and was tested in two experiments - one focusing on classification, the other on diagnosis. The parser was developed so that it could decompose algebraic expressions into a list of terms, each term itself being represented by a list of features (see chapter six). Rather than writing expressions using postfix notation, the parser resolved the problem of representing mathematically equivalent expressions in a single format by generating the ordered string. Again, one library was used for each question type and the data included the set of questions that had been used to train the neural network. The same set of test questions used in the neural network experiment were used to test the system.

7.1.5.1 The Classification Experiment

The first stage of the classification experiment was the construction of a hierarchical domain model. Searching efficiency was improved by imposing a hierarchical structure on the domain knowledge. For the first time, the attribute *Question Type* was to have an abstract representation that was dependent upon more than the keyword alone. Therefore, at the highest (most abstract) level of the domain model was the concept “Algebra Question”, which had two attributes: keyword and expression. At the next level, there were three subconcepts - Expansion, Factorisation and Solve - each of which had its own representation schema (see Figure 7-5).

For example, only expansion problems with a single term were included in the library and were represented by the feature set [operation on bracketed term, number of

[3] CBR-Works is produced by TECINNO GmbH (http://tecinno.com).

subterms, sign, coefficient, number of pronumerals, the list of pronumerals and their powers, number of bracketed terms, the list of bracketed terms and their powers]. The Expansion library was divided into two smaller libraries - one each for multiplication and exponentiation problems. Expansion problems with more than one term were decomposed into a set of problems - one for each term. Factorisation problems were represented by the feature set [GCD_Coefficients, CommonPron, CommonBT, number of terms, term types, list of term signs]. The attributes *CommonPron* and *CommonBT* were Boolean variables that indicated whether there was a pronumeral or bracketed term (respectively) that was common to all terms. This meant that the system could identify both the presence and nature of a common factor. The complete list of *Question Types* can be found in appendix two.

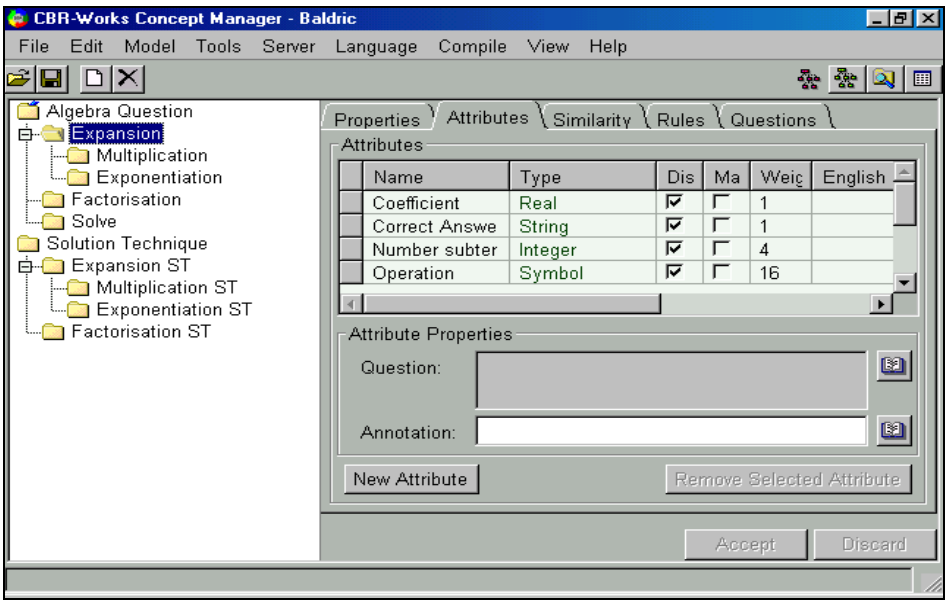

Figure 7-5 Representation schema used in the classification experiment

Aside from the basic structures common to most CBR packages, CBR-Works allows the user to define their own data types and to construct their own similarity measures (using the SmallTalk language). The data types that were constructed are shown in Figure 7-6. Similarity measures depended upon the data types. Those measures that provided greatest discrimination between cases were found to be symmetric quadratic functions for integer types, tables of similarity values for symbol types and exact

string matching for text types. These similarity functions were also used in the final system. The retrieval mechanism used was Nearest Neighbours on the relevant library.

The main aim of the experiment was to determine the weights that should be given to the features used for matching cases and the most appropriate means of assessing similarity and retrieving cases. The algebra problems used as the training set for the neural network experiment were used to construct the first case base, which was tested using the same questions that were used in the neural network experiment. Several technical difficulties were experienced with the construction of queries, which impacted on the testing. However, the limited testing that was conducted on the prototype revealed very good results for the classification task, indicating that the representation schema for algebra questions was suitable for this task.

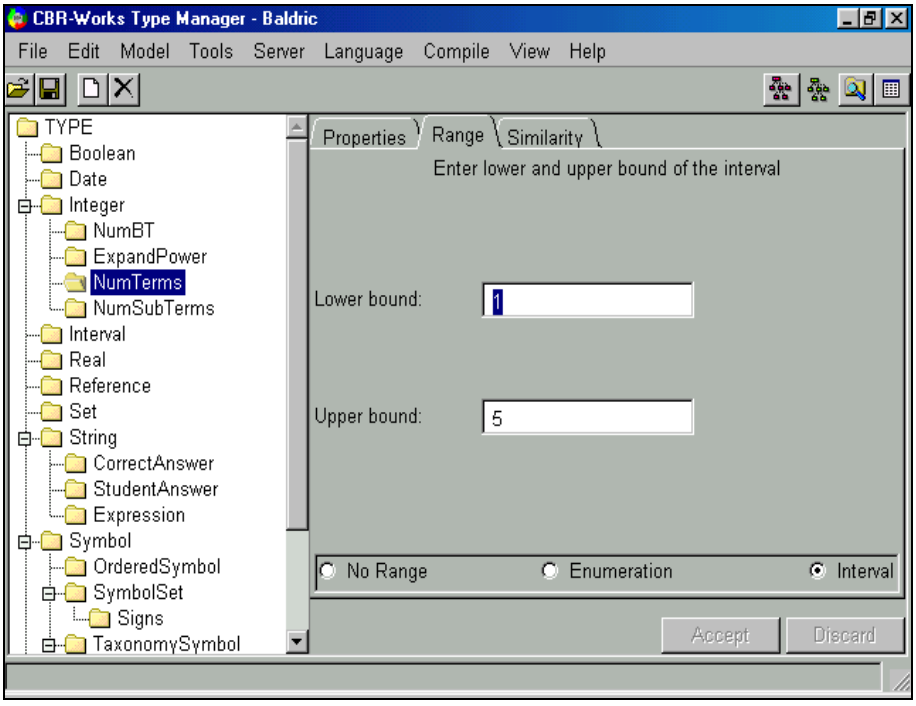

Figure 7-6 The data types used in the classification experiment

7.1.5.2 The Diagnosis Experiment

At this stage, the focus shifted to the diagnostic experiment using the same training and testing sets of algebra questions that were used in the classification experiment.

For each question, the attribute *Question Type* was hand-coded using the output from the classification task. To limit the amount of information that needed to be stored by the system, an exemplar case was created for each value of *Question Type*. These values were derived by using the redundant discrimination networks discussed in chapter three - classification being performed on the basis of the set of solution techniques available. The exemplars incorporated a number of incorrect answers that were derived from standard algebraic errors. These answers were represented by the ordered answer string, the number of terms and the term types. The experiment used the results from the classification task but the focus was now on diagnosis (ie. determining the Solution Technique that led to the incorrect answer by using the **structure** of the answer as a means of matching cases resident in the exemplar file).

Inputs to a query were the question data used for classification, the value of Question Type that had been output from the classification phase, the ordered string corresponding to the "student answer", the number of terms in the "student answer" and the associated term types. The results of the diagnostic experiment were very poor due mainly to the impoverished representation of answers, particularly for questions such as expansion by multiplication where **all** answers have a similar structure. For example, common erroneous answers to a problem such as "*Expand* -2(2*x*-3*y*)" include -4*x*-3*y*, -4*x*+6*y* and -4*x*+3*y*, but the prototype could not distinguish between these because they all have the same structure, viz. number of terms = 2 and term types = [linear_pronumeral, linear_pronumeral].

This was the most important outcome of the experiment. Whilst the number of terms and term types were necessary for discriminating between answers, they were not sufficient. These results raised three questions:

1. What other features are needed to represent answers?
2. How are these features to be extracted from answers?
3. How can the system distinguish between candidate diagnoses?

Resolving these issues led to two decisions. Firstly, the choice of features used to represent an erroneous answer is dependent upon the *Question Type* under

investigation and the errors most commonly made for a given problem type. Those features that provide greatest discriminatory power between cases also provide greatest efficiency and accuracy of retrieval. This led to an analysis of the answers stored in each exemplar. For example, a common error in expansion-by-multiplication problem is incomplete distribution. This occurs when the student only multiplies the first term in brackets by the multiplier (eg. $2(3x-4y+7) \rightarrow 6x-4y+7$). By comparing the list of terms inside the brackets in the question expression with the list of terms in the student's answer, this error can be identified. This analysis led to the final representation schema for answers that was detailed in chapter six.

The second decision was to construct generative mechanisms for each of the solution plans (or algorithms) contained in each of the exemplars. These mechanisms represent the adaptation phase of case-based reasoning, because they represent a solution plan that is instantiated in terms of the current problem, ie. the input is the ordered string that represents the problem expression. The output from a generative mechanism is the ordered answer string for the given solution plan. By using a modular design, generative mechanisms for multi-stage problems could be simply constructed from those for one-step problems (see chapter six). After resolving the issues in matching answers, the final system was constructed. It is now briefly discussed and the results of testing the system are provided in section 7-3.

7.1.6 Final System

Because the technical difficulties that arose with the use of proprietary shells could not be quickly resolved, the final system did not incorporate a shell, instead it was fully coded using KnowledgePro[4] (a development environment incorporating an object-oriented language that allows access to the Windows API). The classification of questions is only required once, so this task is performed at the time of setting a test and the results are stored as part of the test file. This enables the system to use both *Question Type* and *Student Answer* as inputs to the diagnostic phase.

Analysis of the common erroneous answers associated with each case in the *Question Type* case base (see chapter four) led to the identification of the best representation of erroneous answers for each value of *Question Type* (ie. the set of features with greatest discriminatory power). The improved representation of answers led to greatly improved diagnostic performance (see section 7-3). For example, to match answers in the exemplar file for a question of type **EnBm** (multiplying a bracketed term by a signed number) the system extracts the following features:

- *num_terms* - the number of terms in the student's answer,
- *term_types* - the types of terms in the student's answer,
- *numTermsOK* - the number of correct terms in the student's answer,
- *numTermsWrongSign* - the number of terms that have the wrong sign in the student's answer,
- *numTermsInBT* - the number of terms in the student's answer that are also contained in the bracketed term within the question expression,
- *numNegTermsInBT* - the number of terms in the student's answer that are opposite in sign to terms in the bracketed term within the question expression,
- *numCoeffOK* - the number of terms in the student's answer that have the correct coefficient,
- *numPronOK* - the number of terms in the student's answer that have the correct pronumeral list.

For a problem of type **FCFn** (a numerical common factor problem), the system uses the features:

- *num_terms* - the number of terms in the student's answer,
- *term_types* - the types of terms in the student's answer,
- *signOK* - a Boolean that indicates whether the sign on the student's answer is correct,
- *coeffOK* - a Boolean that indicates whether the coefficient on the student's answer is correct,

[4] Knowledge Pro is a product of Knowledge Garden Inc. (http://www.kgarden.com).

- *commonPronOK* - a Boolean that indicates whether the student's answer contains the correct pronumeral common factor,
- *commonBTOK* - a Boolean that indicates whether the student's answer contains the correct bracketed term common factor,
- *nbt* - the number of bracketed terms in the student's answer,
- *numBTOK* - the number of correct bracketed terms in the student's answer.

However for problems that can be solved by a variety of solution techniques, such as a problem of type **FD2Spb** (factorising a difference of two squares problem that has one bracketed term and one pronumeral term) or of type **EPSe** (expanding a perfect square), the features are:

- *num_terms* - the number of terms in the student's answer,
- *term_types* - the types of terms in the student's answer,
- *num_subs* - the number of subterms in each term within the student's answer,
- *coefficients* - the coefficients in each term within the student's answer,
- *signs* - the signs on each term within the student's answer,
- *np* - the number of pronumerals in each term within the student's answer,
- *PL* - the list of pronumerals and their powers for each term within the student's answer,
- *nbt* - the number of bracketed terms in each term within the student's answer,
- *BTL* - the list of bracketed terms and their powers for each term within the student's answer.

These improvements in representation were reflected in the results of the diagnostic phase (see section 7.3 for details). The next section provides details of the needs analysis that was conducted over the life of the research.

7.2 Needs Analysis

In section 2.2, we outlined the need for research aimed at improving cognitive diagnosis. The two main needs were identified as the need for dedicated diagnostic systems that are capable of handling large cohorts of students with a diverse range of abilities in real-time and the need for improved student modelling to support interactive learning environments. The characteristics of useful cognitive diagnosis (Self, 1996) include:

- it should incorporate techniques for hierarchically decomposing domain knowledge from general abstraction at the highest level to detailed forms that can be expanded as required,
- it should be based on a *series* of observations, but should evolve during a session to improve search efficiency,
- it should distinguish between slips (which can be considered to be a failure in execution rather than in intent) and genuine misconceptions,
- it should provide a set of candidate explanations that are ranked in order of likelihood, and
- it should involve the learner interactively rather than passively.

In the remainder of this section, we examine these requirements and explain how they have been addressed in the current system and improvements that could be made in future research. The reader is referred to section 6.5 of the thesis for discussion of the diagnostic output generated by the system.

7.2.1 Hierarchical Decomposition of Domain Knowledge

The first requirement, that of hierarchical decomposition of the domain knowledge, was the most important problem to overcome in the development of an automated system. This has been addressed in the previous chapter, but a summary is included here. Within the system both questions and answers are decomposed by the parser in a manner analogous to that used by computer algebra systems (for example, see Char *et al*, 1991). The tasks performed by the two case-based reasoners are the classification of questions and the diagnosis of errors in incorrect answers. That is, there is a single input to the classifier (an algebra question), whereas the diagnoser has two inputs (the

question and a student's answer). The requirements for representing questions and answers are different (see chapter six) so these are now considered separately.

7.2.1.1 Decomposing an Algebra Question

Questions need to be decomposed before they can be classified. At the highest level, a question is broken into two components: the keyword (which is the goal of the problem and tells the student what action to take) and the expression to be manipulated. The problems associated with representing mathematical expressions as strings were described earlier (see chapter six). Because of the need to maintain mathematical meaning, algebraic expressions are decomposed into a set of signed terms and this set is then ordered by ASCII value. The string that is reconstructed from the ordered set of terms is subsequently used whenever direct string matching is required.

However, question classification proceeds on the basis of the *structure* of a question, so the system also represents the question by a set of features (including the number of terms and the term types). Other features required to represent a question are determined by the keyword and are extracted from the question expression as required. For example, a factorisation problem requires the system to identify factors that are common to all terms, whereas an expansion problem requires the system to identify the operations applied to the bracketed term(s) (see previous chapter for full detail).

7.2.1.2 Decomposing an Answer

Matching answers is dependent upon the *Question Type*, which is the output from the classification phase. The diagnoser uses this output for two tasks; viz. to determine which features to use to represent the student's answer and to open the associated exemplar file. Matching answers is then performed on the basis of the *structure* of answers, rather than by direct string matching. The set of features required for matching is determined by both the problem keyword and the answer form associated

with a particular solution technique. For example, removing a common factor from an expression requires the system to examine the coefficient, sign, pronumerals and bracketed terms contained in each term of the student's answer to determine whether the factor was removed from every term, whereas expanding a perfect square focuses on determining both the number and types of the terms in the student's answer.

In summary, the system represents both algebra questions and student answers in an abstracted form, this form being determined in most part by the goal of the problem. This is reflected in the structure of the case libraries (see chapter five).

7.2.2 Ranked List of Proposed Diagnoses

A second requirement of useful cognitive diagnosis is the provision of a **list** of possible diagnoses, where this list is rank-ordered by likelihood. Initially the case-based diagnoser generates a hypothesis list of diagnoses. This list is rank-ordered in accordance with the similarity scores between the student's answer and each of those contained in the exemplar file. However, it is common to find that several solution techniques lead to the same answer (and hence the same answer structure). The system therefore requires a means of distinguishing between the candidate diagnoses; this work is performed by the generative mechanisms (see section 6.4.4), which represent the adaptation phase of case-based reasoning.

Each candidate diagnosis is associated with a generative mechanism, ie. the algorithm (or solution technique) that produces the associated answer in the exemplar file. These programs are instantiated in terms of the current question expression (ie. the input is the problem expression string) and output an ordered answer string (ie. the output represents the answer generated by the associated solution plan for the problem under investigation). During diagnosis of a particular erroneous answer, the system runs the generative mechanisms for each of the solution techniques appearing on the hypothesis list. Each generated answer is then compared with the student's answer using direct string matching. An exact match will occur if the system has correctly determined each step in the student's solution plan, but can also occur when

alternative solution paths lead to the same answer. For this reason, the student needs to be consulted to determine the validity of the diagnosis. This is now discussed further.

7.2.3 Student Involvement

Educationalists have stressed the need for students to be involved in diagnosing their own weaknesses and misconceptions in order that they develop correct knowledge structures and checking strategies (Rosnick and Clement, 1980, Ginsburg, 1983, Davis, 1984, Schoenfeld, 1985, 1987, Tall, 1991, Perrenet and Wolters, 1994, and Chi *et al*, 1988). This was reinforced by Self as a criterion for useful cognitive diagnosis (Self, 1996). Our system provides limited student involvement.

As outlined above, once the system has created a list of potential diagnoses, it uses the generative mechanisms to find one or more exact matches for the student's answer. A list of diagnoses which lead to the same answer as the student's is then produced. If the list is not empty, then the associated diagnoses (which include the step-by-step working produced by the generative mechanism) are presented to the student who can either accept or reject the diagnosis. However if the list is empty, then the student's answer may need to be added to the appropriate exemplar file (see next section for more detail). This requires the system to engage in a dialogue with the student to elicit information about their solution technique. Methods for implementing dialogues with the user are an active area of research (for example, Moore, 1995a, 1995b, and Kashihara *et al*, 1995), but have not been implemented here. Future work in this area of cognitive diagnosis will investigate the most appropriate means of handling student interaction within the diagnostic system.

7.2.4 Set of Observations

Learners have a number of characteristics including incomplete, incorrect and poorly-connected knowledge schemata, impoverished problem representations and imperfect concepts of similarity. For these and other reasons such as forgetting, a problem that is

not encountered in machine learning, human problem-solving behaviour is notoriously unstable (Brown and Van Lehn, 1980, Van Lehn, 1982, 1983, Booth, 1984, Davis, 1984, Smith, 1987, Schoenfeld, 1987, Siegler, 1987a, 1987b, 1988, Van Lehn, 1989, Vinner, 1990, Pierce and Gholson, 1994, Hampson and Morris, 1996, Adibnia and Putt, 1998). The error analysis conducted at the outset of the current research resulted in similar findings (see chapter four). These observations led Self to recognise that cognitive diagnosis is much more difficult to realise than fault diagnosis in machinery, and that cognitive diagnosis should therefore not be based upon a single observation, but rather should evolve throughout a session and should involve a number of related observations (also see Birenbaum *et al*, 1993).

The current system only partially realises this aim. For each question that is incorrectly answered, the system attempts to diagnose the errors and uses a measure of the student's performance to guide the search (see previous chapter). However, complete diagnosis should not only identify the solution technique that a student applied to an individual problem, but should identify when there is a change in problem-solving performance and which techniques are typically preferred by the individual. This is related to the problem of identifying whether a particular solution method represents one that the student genuinely believes to be correct and appropriate or whether it simply represents a work-around when the student can not progress with solving a problem. This is now discussed in more detail.

7.2.5 Misconceptions, Slips and Repairs

The use of multiple observations in cognitive diagnosis can provide help in determining whether an incorrect answer is the result of a slip in the execution of a valid technique, a genuine misconception or the use of a "repair" (ie. a method that enables the student to work around an impasse or obstruction to problem solving (Van Lehn, 1983)). The question to be answered is "Does an adopted 'solution technique' represent a method that the student genuinely believes to be valid or is it simply a repair?". It is important to distinguish between the possibilities so that remediation and further teaching are directed appropriately.

At present, the diagnostic system does not attempt to answer this question; it simply aims to identify the solution technique applied to each question and to reconstruct the student's line-by-line working. However, the data that are collected for each test item could be collated to form the basis for extended post-test analysis with the aim of identifying *patterns* in the individual's problem-solving behaviour (ie. identifying a student's knowledge state in terms of preferences for particular solution techniques over others, the stability of these preferences and the level of problem detail that induces a change in the choice of strategies).

In this section, several needs required for cognitive diagnosis, and the extent to which they have been realised in the current system, were addressed. In summary, the system is capable of decomposing questions and answers using domain knowledge and returns an ordered list of hypothesised diagnoses. Other characteristics of useful cognitive diagnosis (viz. student involvement and the use of multiple observations to help in distinguishing between genuine misconceptions, slips and repairs) are not fully realised. These are to be the subject of future research. In the next section, we evaluate the system's performance in its two main tasks, viz, the classification of algebra problems and the diagnosis of errors.

7.3 Performance Measures of the Final System

The previous section provided details of the evolution of the diagnostic system, whilst this section focuses on the performance of the final system in its dual tasks of problem classification and error diagnosis. Evolution was conducted as a series of experiments, where the aim was to determine how data should flow through the system, how cases should be represented, how to implement similarity calculations, the most appropriate means for retrieving cases and the structure of the case libraries. Details of system implementation were provided in chapter six and the performance of the system is now detailed.

A useful case-based system is one that retrieves all relevant cases and does this as efficiently as possible. Evaluation of system performance therefore focused on the following questions:

1. Does the system retrieve the most relevant case first?
2. Does the system retrieve all useful (ie. relevant) cases?
3. Does the system only retrieve useful cases?
4. How efficient is the system?
5. How consistent is the system? This is important here because whilst there are finite numbers of problem types and answer structures, there are infinitely many variations on the theme.

These questions were distilled into three measures of performance: accuracy, efficiency and consistency. The performance of the two case-based components (the classifier and the diagnoser) are now discussed in terms of these three measures.

7.3.1 Performance of the Classifier

To determine the optimal feature weights, the final system was trained using a set of 198 algebra problems (97 expansion problems and 101 factorisation problems). This set was chosen to include at least three examples for each case within the *Question Type* case library, including the NULL cases. The features required for matching problem types had already been determined, but the final weights that are assigned to particular features required fine-tuning. The decision was made that factorisation problems that contained a numerical common factor but also matched another standard form (such as General Quadratic) should be classified as the more specific type. Feature weights were adjusted to enable this and the exemplar problems that were chosen to represent the specific types included a numerical common factor. On the other hand the recursive nature of mathematical problem solving means that when an algebraic common factor has been removed from a problem, a subproblem of a new type may emerge. This requires a human problem solver to consider the emergent subproblem, categorise it and determine a suitable solution strategy. However, the

system currently only categorises the complete problem at the time of setting; future research will address this limitation (see section 7.6).

The case libraries contain 24 expansion and 37 factorisation problem types respectively (including NULL). To test the system's performance in classifying algebra problems, a new set of 279 problems, containing 134 expansion problems and 145 factorisation problems, was generated. For each *Question Type* contained in the case libraries, a minimum of 5 examples was included, and at least 5 examples were included for each compound factorisation problem type (ie. those that can be correctly classified on structure but also contain a numerical common factor). As well, the system was tested using some problems that do not exactly fit any cases in the libraries, eg. the problem "Expand $(a+b)(c+d)(e+f)$" contains three bracketed terms multiplied together, which, to be consistent with the existing types, would have type EB3m, however this case is not included in the library. The aim of including such problems was to assess whether the system would retrieve the cases that were anticipated to be the nearest match, or whether the modelling of the classification task required dramatic changes before it could be extended to cover other problem types.

7.3.1.1 Accuracy of Classification

Two experienced secondary-level teachers[5] were involved throughout the testing of the system. Their task was to assess whether the system's output was consistent with the discrimination networks, and also to identify irrelevant cases that were returned and relevant cases that were not returned. Classification of the 134 Expansion problems returned a total of 424 hypothesised types, whilst the classification of the 145 Factorisation problems returned a total of 374 hypothesised types. Accuracy of classification was measured in three ways: correctness, precision and recall. *Correctness* refers to whether or not the system assigns the highest similarity value to the most on-point case. The second measure, *precision*, is defined as shown in Figure 7-7.

$$precision = \frac{number\ of\ relevant\ cases\ found}{total\ number\ of\ cases\ found}$$

Figure 7-7 Measuring relevance of retrieval

That is, *precision* is a measure of the relevance of the set of cases retrieved by the system. The final measure, *recall*, is defined as shown in Figure 7-8.

$$recall = \frac{number\ of\ relevant\ cases\ found}{number\ of\ relevant\ cases\ in\ library}$$

Figure 7-8 Measuring the completeness of retrieval

That is, *recall* is a measure of the completeness of retrieval (see Wallis and Redding, 1996).

For expansion questions, the system had a *correctness* value of 94%, ie. for 8 of the 134 problems in the set, the classifier determined that the problem was most probably of a type other than the most on-point case. These problems included two cases where only one problem in a multi-term problem was classified, two cases of NULL problems that were classified incorrectly and the four cases that had type EB3m. In each of these instances, the system retrieved three cases; in order of similarity value, these were EB2m, EpB2m and EpBm. These queries do not really represent errors, since there is no exact match in the case library, but they do indicate that the modelling of the classification task should be easily extended to incorporate new problem types. The initial similarity function used for comparing the number of bracketed terms in an expression was a step function; this was replaced by a symmetric quadratic function. Once this adjustment was made to the similarity function, the problem was corrected without altering the classification output for other expansion problems.

For factorisation questions, the system had a *correctness* value of 93%, representing errors in classifying 9 problems. These problems included one three-term General

[5] My thanks to Bronwyn Barlow and Leone Walbran for their enthusiastic involvement.

Quadratic problem that was classified as a Grouping problem and the remaining cases were all problems that were classified as General Quadratic when other, finer classifications (such as Difference of Two Squares) were expected. In each of these eight cases, the correct classification was retrieved as the second most probable case. This situation led to a slight modification of feature weights that subsequently resulted in the problems being correctly classified.

For expansion questions, the system retrieved a total of 424 cases, of which 302 were deemed to be relevant by the two school-teachers involved in evaluating system performance. That is, the system had a *precision* value of 71% when classifying expansion problems. The problem types for which the system retrieved additional cases included EnBm (problems where a bracketed term is to be multiplied by a number), EnpBm (problems where a bracketed term is to be multiplied by both a number and pronumeral terms), and EnPSe (problems where a bracketed term is to be squared and then multiplied by a number). The extra cases that were returned by the system were EnpBm and EnB2m for EnBm problems, EnpB2m for EnpBm problems, and EnpPSe and EpPSe for EnPSe problems. Because these additional cases were not required during the diagnostic phase, no adjustments were made to either the feature weights or the similarity functions.

For factorisation questions, the system retrieved a total of 374 cases, of which 298 were deemed to be relevant. That is, the system had a *precision* value of 80% when classifying factorisation problems. The problem types for which the system retrieved additional cases included FGQ3np (general quadratic problems containing three terms that are either pronumeral terms or a number), FGroup3 (problems containing three terms that require the solver to group two of these terms and search for a common factor) and FGroup4 (problems containing four terms that require the solver to group the terms and search for a common factor). The extra cases that were returned by the system were EnpBm and EnB2m for EnBm problems, EnpB2m for EnpBm problems, and EnpPSe and EpPSe for EnPSe problems. Because these additional cases were not required during the diagnostic phase, no adjustments were made to either the feature weights or the similarity functions. The extra cases that were returned by the system

were FGroup4 for FGroup3 problems, FGroup3 for FGroup4 problems, and FGroup3 for FGQ3np problems. Again these additional cases were not required during the diagnostic phase, and hence no adjustments were made to either the feature weights or the similarity functions.

For expansion questions, the system had a *recall* value of 77%. The problem types for which it was expected that the system would retrieve additional cases were EnpB2m (problems where the product of a pair of bracketed terms is to be multiplied by both a number and pronumeral terms) and EnpBm problems (where a bracketed term is to be multiplied by both a number and pronumeral terms). For the first problem type, the system retrieved EnB2m, but not EpB2m, and similarly for the second problem type, the system retrieved EnBm, but not EpBm. In each situation, the system assigned greater similarity scores for exponentiation problems than for the missing category. Either of two methods could be used to address this problem, viz. assigning a greater-than-current weight to the problem feature *Operation on Bracketed Term*? or separating the single *Question Type* case library into two libraries, one each for exponentiation and multiplication cases.

For factorisation questions, the system had a *recall* value of 89%. This means that the system failed to retrieve some cases that were deemed to be "close enough" to the query case to qualify for retrieval. All of these cases were problems that could be completely solved by extracting a common factor and they all returned similarity scores of 100% for the relevant category, but no other cases were retrieved. For example, a problem that can be correctly classified as having type FCFnpb (ie. a problem for which all terms contain a common pronumeral term, a common bracketed term and for which the greatest common divisor of the coefficients is greater than one) could also be expected to generate high similarity values for the cases FCFn, FCFp, FCFb, FCFnp, FCFpb and FCFnb. The reason that none of these cases was retrieved is because the features required for identifying each type of common factor are evaluated using similarity functions that include penalties when there is no match on a feature. In fact, the task of identifying and classifying a common factor could be implemented

simply by using rules. This suggests that future research should address the issue of adopting a hybrid approach to implement the classification task.

The results of the classification task are summarised in Table 7-5. The second element of system performance testing, viz. its efficiency, is now detailed.

Table 7-5 Results of testing the classifier

Measure	Expand	Factorise
Correctness	94%	93%
Precision	71%	80%
Recall	77%	89%

7.3.1.2 System Efficiency in Classification

System efficiency has been obtained by basing the structure of the case libraries on the value of the problem keyword. The classifier thus takes the same time to classify each expansion problem. This is because the system searches the entire expansion library using a nearest-neighbours approach to calculating the similarity scores, and search time is therefore determined by the size of the case library. The situation is similar for factorisation problems, although these take a longer time to classify than expansion problems, due to the larger case library needed to represent factorisation problems. Improvements in system efficiency could be realised by improving the library structures for each problem type or by adopting a hybrid approach to classification, rather than a purely case-based approach (see section 7.3.1.4).

7.3.1.3 System Consistency in Classification

Because the system is completely automated, one of the most important aspects of its performance in classifying algebra problems is its consistency, ie. are problems of the same type always classified the same way by the system? Consistency is important because, although there are only finitely many types of problems, there are infinitely many variations available for each type. On the other hand, the domain of algebra is

very well-defined; that is, it is possible to define problem types in such a way that either a human or the system can uniquely determine the type of an algebra problem. It is for this reason that the classification task is based upon the **structure** of the problem expression, rather than on surface features such as the values of coefficients, pronumerals, bracketed terms and signs (except for cases where these are relevant, such as distinguishing between a Difference of Two Cubes problem and a Sum of Two Cubes problem).

To test system consistency, each problem type was represented by at least six examples. It was found that the system behaved completely consistently when classifying algebra problems. As an example, consider the problem "*Expand* $-2a(m+n)$", which has type EnpBm. Other examples that the system correctly identified as being of this type included "*Expand* $7.4*a^2*c*(9*m^2-u)$" and "*Expand* $-4*b^2*c^8*(6.7*m^2*(x-z)-3*h*u^3)$". Not only did the system recognise that these problems had the same type (that is, the most on-point case was assigned the highest similarity score), but the system returned identical ranked hypothesis lists in each instance. Similar results were obtained for the set of factorisation problems used to test the system.

Whilst the above results indicate that the system performs well at the classification task, a number of improvements will be investigated in future research. These improvements are outlined next.

7.3.1.4 Future Improvements

Exhaustive testing revealed that the set of cases retrieved by the system always contained the most on-point value for *Question Type*, although this may not have been the case with the highest similarity value, and in some cases provided alternatives. Library structure affects the performance of the classifier. Current structure is shown in chapter six, but improvements in performance could be realised by making several changes as indicated in Figure 7-9.

At present, the attribute *Question Type* for expansion problems can be uniquely determined using the discrimination network shown in chapter six. However, improvements could be made in system efficiency by splintering the Expansion library into two: one each for multiplication and exponentiation problems, and to have the system use the value of the attribute *Operation on Bracketed Term*? to determine which of the two libraries to search.

However the classification of factorisation problems based on the structure of the initial problem expression can only identify the *Question Type* at the highest level because, whilst the presence of a numerical common factor does not mask the structure of a problem, the presence of an algebraic common factor can partially mask the problem structure. For example the problem "*Factorise* $2a^2 + 6a + 4$" is both a numerical common factor problem and a general quadratic problem. The system recognises this and returns both values from the classification task, whereas the problem "*Factorise* $3a^2b^3 + 24a^2c^3$" would only be classified as being an algebraic common factor problem. This is because the system currently does not classify subproblems which emerge during the solution process. However for this example, once the common factor ($3a^2$) is removed from the problem expression, a new subproblem "*Factorise* b^3+c^3" ,which has *Question Type* Factorise Sum of Two Cubes, emerges. Thus, the second change that is required for improved classification performance is more complex than the first. To improve the classification of factorisation problems, the system would need to divide the factorisation library in two, with separate libraries for Common Factor problems and other factorisation problems. The features *GCD_Coefficients*, *CommonPron* and *CommonBT* could serve to flag problems that contain a common factor.

To address this in future, the first step is to expand the discrimination network for factorisation problems and then to expand the functionality of the classification task by introducing recursion:

1. Determine whether there is a common factor (using current feature set as flags). If not classify the problem using the library of *Other Factorisation Problems* and stop.
2. Determine the complete common factor (using the features *GCD_Coefficient*, *CommonPron* and *CommonBT*).
3. Divide the original question expression by the common factor (using a computer algebra system) to obtain the emergent subproblem.
4. Classify the subproblem.
5. These changes will impact on the diagnostic task. The system will need to use the structure of the student's answer to determine whether the complete and correct common factor has been removed. If not, the student may not be able to recognise the new subproblem and their answer should be diagnosed using the relevant Common Factor exemplar. If the student has correctly identified the complete common factor AND has performed some other steps in their solution plan, then the exemplar for the new subproblem would also need to be investigated.

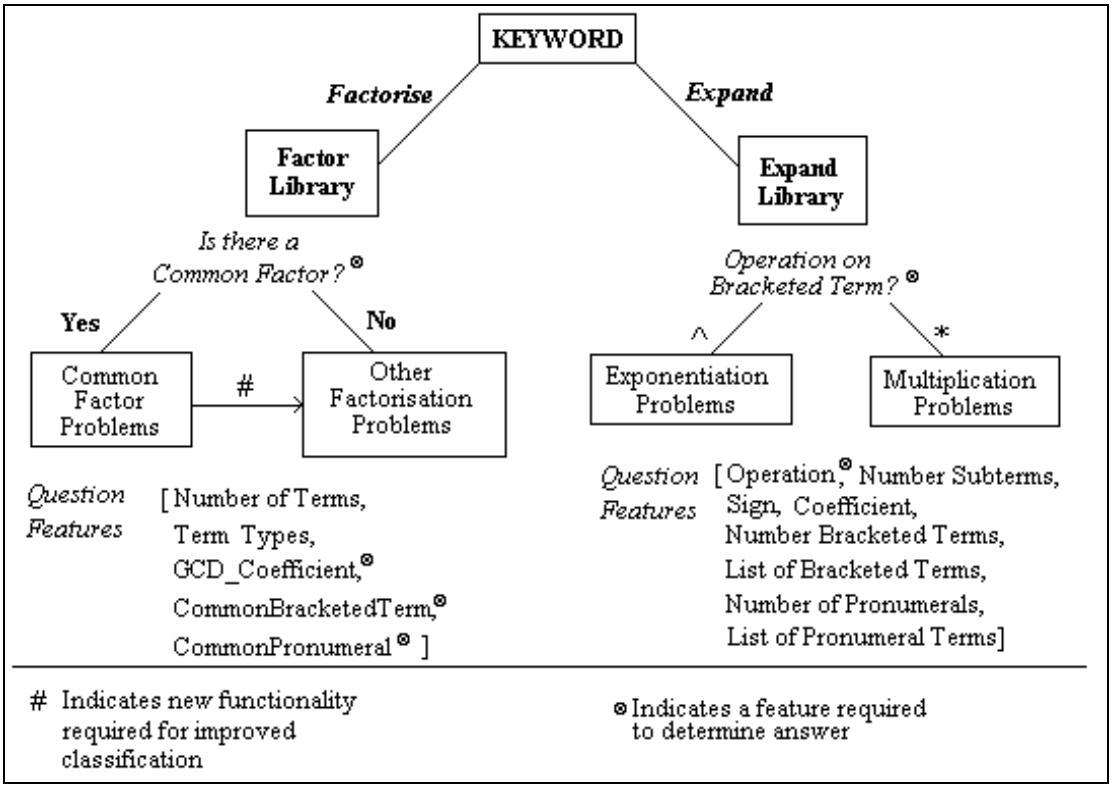

Figure 7-9 Proposed improvement to library structure

In summary, the results of testing the classifier are shown in Table 7-5. They show that the system can classify algebra problems with a high level of accuracy. This was to be expected because of the hierarchical nature of the algebra domain and indicates that discrimination networks provide a useful means of representing the task of classification. This was important because the classification task provides the link between an algebra question and a student's answer. The performance of the diagnostic component of the system is detailed in the next section.

7.3.2 Performance of the Diagnoser

The diagnosis of errors is the main task performed by the system. To achieve this, the diagnoser acts in the following manner:

1. Using the hypothesis list returned from the classification task, identify the *Question Type* with the highest similarity score.
2. Open the exemplar file associated with the given value of *Question Type*.
3. Use the value of *Question Type* to determine the feature set required for matching answers in the exemplar case file.
4. Extract the feature set from the query case.
5. Generate similarity scores between the student's given answer (the query) and all cases contained in the exemplar file, using Nearest Neighbours calculations.
6. Rank-order the list of similarity scores and return a list of the candidate cases that have a similarity score above the threshold value.
7. Run the generative mechanisms for cases on the candidate list to validate the proposed diagnosis. (Not all cases in the exemplars have generative mechanisms implemented, which means that the system is not always able to validate a particular diagnosis. It was felt that this would be helpful because it would enable testing of cases involving such exemplars to focus on other aspects of answer matching.)
8. Choose the case that is most on-point; this will be the case whose generative mechanism led to the same answer as that input by the student (if such a case

exists) or the case with an answer structure that is closest to that of the student's answer.

It is important that the system diagnoses answers both accurately and consistently; ideally the system should also be as efficient as possible. Therefore, the same three performance measures that were applied to the classifier (viz. accuracy, efficiency and consistency) were also used to evaluate the performance of the diagnoser. The data used to test the diagnoser included problems of each question type, answers corresponding to each answer contained in the exemplars and some answers that did not correspond to any known cases. To test the system's consistency, several problems were generated for each question type. A single method (eg. Generalised Distributivity) was used to generate the answer to each problem in the set to investigate whether the system would retrieve the same diagnosis. The "answers" were generated in collaboration with two secondary teachers, because it was felt that they would be familiar with typical errors made by students (this is similar to Virvou and Tsiriga, 1999). A total of 429 question-answer pairs were tested; the results of this testing are now detailed.

7.3.2.1 Accuracy of Diagnoses

Testing comprised the diagnosis of 429 question-answer pairs. The system retrieved a total of 1408 cases; of these, 924 cases were deemed to be relevant. The number of cases in the libraries that were deemed to be relevant totalled 979. In 22 cases, the system could not diagnose the errors (representing a failure rate of 5%). A further 55 cases, or 13%, could be only partially diagnosed. For the remaining cases, accuracy of diagnosis was measured in three ways: *correctness* (ie. whether the most on-point case had the highest similarity score), *precision* (ie. whether all the retrieved cases were relevant) and *recall* (ie. whether all relevant cases were retrieved).

For 264 of the queries, the system assigned the highest similarity score to the most on-point case, representing a correctness value of 62%. Precision was assessed as 64% (of the 1408 retrieved cases, 924 were relevant) whilst recall was measured at 94%

(the library contained 979 relevant cases and 924 of these cases were retrieved by the system). The question types for which the system could not completely and correctly diagnose errors were confined to some expansion by multiplication problems (for example, the problems "*Expand* $(2a-b)(3b+4a)$" and "*Expand* $-2x(1-3x)$") and factorisation problems that contained an algebraic common factor that disguised the presence of a subproblem (for example "*Factorise* $2ab^2-8a(m-n)^2$" which contains the common factor *2a*; once this factor is removed the new problem "*Factorise* $b^2-4(m-n)^2$ emerges"). These issues are now addressed.

The expansion problems that led to an incomplete diagnosis were of two main types; the first of these was EnpBm problems (ie. those problems that require the student to multiply a single bracketed term by both a signed number and a pronumeral expression) where the single bracketed term includes terms containing (some of) the same pronumerals in the multiplier. The generative mechanisms for these two problem types have not been implemented. An example is the problem "*Expand* $-2x(1-3x)$"). The answer that was undiagnosed was *-x*, which was obtained in the manner shown in Figure 7-10.

$-2x(1-3x)$
$\rightarrow 2x-3x$ (*wrong sign on first term AND incomplete distribution*)
$\rightarrow -x$ (*correct collection of like terms*)

Figure 7-10 Working that could not be diagnosed

As noted, the system was unable to identify the method that resulted in this answer. However, when the input to the diagnoser is the expression "*2x-3x*", the system correctly identifies the student's solution technique and retrieves the appropriate case. The reason for the system failing to diagnose this answer are two-fold: firstly, the exemplar problem is "*Expand* (3**a*-2**m*)*4**c*" (where the bracketed term does not have any pronumerals in common with the multiplier) and, secondly, whilst the exemplar does contain a case corresponding to the erroneous answer, the generative mechanism has not been implemented. Future work will investigate two possible

remedies. The first of these is to implement the generative mechanism for the closest matching case (ie. the expression "*2x-3x*",) and then to add an extra case to the exemplar that corresponds to the given answer (this simply requires expanding the existing generative mechanism by adding a call to *Collect Like Terms*). The second method entails choosing a new exemplar that does have pronumerals inside the bracketed term that are common to those in the multiplier.

The second set of problems for which the system could not completely diagnose the errors were problems of type EB2m (where two bracketed terms were to be multiplied together) and for which the student adopted the following variation of the FOIL technique:

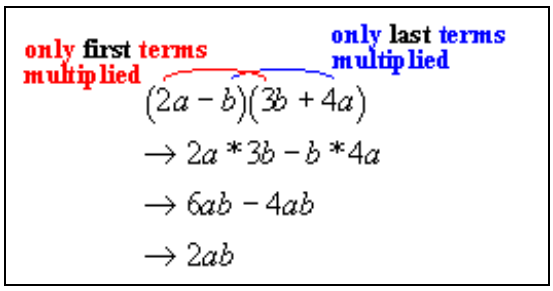

Figure 7-11 An example of an erroneous application of the FOIL technique

Again, the exemplar does contain a case that corresponds to the second last line of the student's working (viz. "6*ab* - 4*ab*"). However, the problem used as the exemplar contains four terms, none of which share a common pronumeral. This problem is very similar to that for the EnpBm problem discussed above and hence will be addressed at the same time.

The final problem type for which the system could not fully diagnose an incorrect answer was the set of problems that were classified as having an algebraic common factor and where the common factor disguises the presence of a subproblem. An example is the problem "*Factorise* $2ab^2-8a(m-n)^2$". The student's working for this problem is shown in Figure 7-12.

$$2ab^2 - 8a(m-n)^2$$
$$\rightarrow 2a\left(b^2 - 4(m-n)^2\right)$$
$$\rightarrow 2a\left(b + 4(m-n)\right)\left(b - 4(m-n)\right)$$

Figure 7-12 Student working for a compound factorisation problem containing a common factor

Under the current implementation, classification takes place in a single pass, that is, once the system has classified the problem as having an algebraic common factor, it cannot identify and classify subproblems that emerge during the solution process (see previous section). Given that the system cannot fully classify such a problem, it is therefore impossible for the system to diagnose answers to it. In section 7.3.1.4, improvements to future versions of the system aimed at improving the system's performance during the classification task were outlined. It is expected that once this is achieved, system performance in diagnosing erroneous answers will also be improved. We now discuss the efficiency if the system during its diagnostic phase.

7.3.2.2 System Efficiency in Diagnosis

The system's efficiency in diagnosing the errors in a particular question-answer pair is determined by the structure of the libraries, the success of the system in classifying the problem, the size of the relevant exemplar case library and the number of generative mechanisms that must be run to distinguish between candidate diagnoses. Currently, the system is faster in diagnosing errors for problems that have available solution techniques that lead to answers that have significantly different structures. It is less efficient in diagnosing errors on procedural problems (those that have limited solution techniques available and where there is little difference in the structure of the resultant answers) for example expansion-by-multiplication problems such as "*Expand* - 5*(3a^2b-7cd^3)". However, the exemplar files tend to be quite small (at present, the largest exemplar file contains 15 answer cases), and the use of generative mechanisms to distinguish between candidate diagnoses is proving to be the most effective means of diagnosing errors on these problem types.

On the other hand, diagnosis time for answers to more complex problems could be improved by including a layer of abstracted answers within the exemplar file. For example, consider the problem type FD2Spb (which is a Difference of Two Squares problem that contains a quadratic pronumeral term and a quadratic bracketed term of opposite sign). Currently, the exemplar contains 15 cases corresponding to erroneous solution techniques. Of these cases, three pertain to the *Solution Technique* Generalised Distributivity, two pertain to template methods, two cases to the *Solution Technique* Expand and Factorise (using a template for the expansion subproblem) and five cases to *Solution Technique* Expand and Factorise (using Generalised Distributivity for the expansion subproblem). Because related solution techniques give rise to answers that have similar structure (number and types of terms), the cases could be clustered by structure, which would reduce the number of cases for matching to six. Once the abstracted case that is the closest match to the query has been retrieved, all cases that belong to the cluster would be retrieved and either the associated generative mechanisms or other features extracted from the student's answer could be used to distinguish between candidate diagnoses in the set.

The last aspect of system performance, consistency of the diagnoser, is now discussed.

7.3.2.3 System Consistency in Diagnosis

Apart from accuracy, the most important aspect of the diagnoser's performance is consistency. To evaluate this, the system was tested using fifteen sets of question-answer pairs. Each set comprised problems that were of the same type and each answer to the problems in a single set were generated using one of two solution techniques - one for which there was an exact match within the exemplar and one for which there was no exact match.

For the queries where there was an exact match in the exemplar file, the system consistently retrieved the most on-point case. Examples included here were expansion-by-exponentiation problems solved using Generalised Distributivity (for example, "*Expand* $(7s-6h)^3 \rightarrow 343s^3 - 216h^3$"), expansion-by-multiplication

problems solved using Incomplete Distribution (for example, "*Expand* $-5(7s-6h) \rightarrow -35s-6h$"), pure common factor problems that had been incompletely factorised (for example, "*Factorise* $7+7b^2+14 \rightarrow 7(1+7b^2+14)$") and factorisation problems that did not contain a common factor and that were solved using a known erroneous method (for example, "*Factorise* $c^2-(6h+4b)^2 \rightarrow c^2-36h^2-16b^2$").

On the other hand, when the system could not find an exact match, it retrieved a **set** of cases from the exemplar, each of which was assigned the same similarity score. Examples of question-answer pairs included expansion-by-exponentiation problems solved using a variation of Generalised Distributivity that was not included in the exemplar file (for example, "*Expand* $(7s-6h)^3 \rightarrow 21s^3-18h^3$"), expansion-by-exponentiation problems for which there was no exact match in the exemplar (for example, "*Expand* $(a+2b)^3 \rightarrow a^3+2a^2b+4ab^2+6b^3$") and compound factorisation problems (that is, a factorisation problem that contained a common factor and for which the structure also matched another question type, for example, "*Factorise* $50n^2-2d^2 \rightarrow 2(25n^2-d^2)$").

For these queries, the system varied in its ability to distinguish between the candidate diagnoses. For the set of queries that included the example "*Expand* $(7s-6h)^3 \rightarrow 21s^3-18h^3$"), the system assigned the highest similarity score to all four cases corresponding to the *Solution Technique* Generalised Distributivity within the appropriate exemplar, but could not discriminate between them; ie. it could not determine the manner in which the coefficients were calculated. It was also found that for this set of queries, the system ranked all cases in the exemplar in a consistent manner. That is, the query above returned an identical vector of similarity scores as did the query "*Expand* $(2ab-7h)^3 \rightarrow 6a^3b^3-21h^3$". Similarly, the query "*Expand* $(a+2b)^3 \rightarrow a^3+2a^2b+4ab^2+6b^3$" returned an identical vector of similarity scores

as the query "*Expand* $(a+7)^3 \rightarrow a^3+7a^2+14a+21$", whilst the query "*Factorise* $50n^2-2d^2 \rightarrow 2\left(25n^2-d^2\right)$" returned an identical vector of similarity scores as the queries "*Factorise* $7a^2-63w^2 \rightarrow 7(a^2-9w^2)$" and $3a^2-12(b-7u)^2 \rightarrow 3(a^2-4(b-7u)^2)$.

In summary, the diagnoser does perform consistently. This is desirable and is to be expected because of the nature of the domain knowledge. For a set of queries that represent questions of the same type and answers generated by a single solution technique, the system always returns the same vector of similarity scores for cases within the relevant exemplar. The generative mechanisms improve the discriminatory power of the system when several candidate diagnoses return identical similarity scores.

7.3.2.4 Future Improvements in Diagnosis

Several issues are to be addressed in future research to improve the diagnostic capability of the system. These were discussed above and include:

1. restructuring case libraries to improve efficiency,
2. implementing generative mechanisms for every case in each exemplar file,
3. reviewing the choice of exemplar problems, and
4. restructuring the Factorise case libraries and the method of classification to improve the system's diagnostic capability for compound factorisation problems.

Other issues that will be addressed in future include the need to choose the implementation paradigm to fit the situation. That is, it is felt that it is unlikely that optimal diagnostic performance will be achieved using a single means of implementation. Instead a hybrid diagnoser that includes (at least) both rule-based and case-based reasoning components could possibly outperform the existing system, because the answers to problems at different levels of complexity require different means of representation, indexing and matching. This is related to the choice of exemplar problems. These issues are discussed in more detail in section 7-6.

In summary, the results of testing the diagnoser are shown in Table 7-6. These results indicate that the system recalls a high percentage of the relevant cases. In many instances, several queries returned a set of cases with the same similarity measures, thereby reducing the values for the measures of correctness and precision. However, the inclusion of generative mechanisms has enabled the system to distinguish between solution techniques that lead to answers with the same structure. In the next section, the issue of system maintenance is discussed.

Table 7-6 The results of testing the diagnoser

Measure	Value
Correctness	62%
Precision	64%
Recall	94%

7.4 Maintenance of the System

The main purpose of the system is the diagnosis of errors on algebra questions. Hence, the focus of the research to date has been on the design and development of the basic system, which has not been tested with large numbers of potential users. The research has involved two experienced secondary-level teachers for the purpose of evaluating the validity of the model of mathematical problem solving as well as the completeness and correctness of the diagnoses returned by the system and advising on the role that such a system could play within the classroom. However, maintenance is fundamental to the ongoing use of any system and this section discusses some pertinent issues that will be addressed in future research.

It is anticipated that the diagnostic system will require significant maintenance to optimise its performance. Because of the well-defined nature of the algebra domain, it

is expected that the existing *Question Type* case library will require little or no modification. This expectation is supported by the results of evaluating the classification task, which showed that the system is both accurate and consistent in its performance, indicating that case coverage is close to complete. However, it is anticipated that the *Solution Technique* case library will require significant maintenance. Students, particularly early learners, can be very creative in producing their own solution techniques, so it is important that we identify those new methods that are worth storing. Repair theory states that students may produce new solution methods when they reach an impasse during problem solving (see Brown and Van Lehn, 1980). However, unless the method is repeated, it is more likely to represent a patch than a genuine new technique. This is the criterion used to determine whether or not to store a new case in the *Solution Technique* case library.

Similarly, it is expected that future students will produce new errors in executing a single step of a standard solution plan. Again, repetition of the error on a complete test will be the best indicator of whether or not the case needs to be added to the library. For example, suppose that a student is answering the problem "*Factorise* $16x^2 - (2b-7)^2$" by applying the *Solution Technique* Expand and Factorise, that the student makes an error in expanding the squared term and that this error had not previously been encountered by the system. If the decision is taken to incorporate the error case in the system, then **two** cases must be added - one is the new solution case to be added to the exemplar for the complete factorisation problem, and the second is the complete case to be added to the exemplar for the expansion problem. The latter case requires a generative mechanism to be encoded; this mechanism can then be called, as required, by the solution plan for the first case.

The issue of how maintenance will be best achieved is problematic; however, it is certain that it will not be fully automated. The basic elements for storing a new case in an exemplar file are the question, its question type, the correct answer, the student's answer, the relevant set of features needed to represent the answer, a text file containing the explanation of the student's solution and the generative mechanism.

The first five elements can be obtained automatically, but the other two require human intervention. Firstly, the steps in the student's solution technique must be identified, which will involve input from a mathematics education expert. Two possible methods for realising this on-line are conducting an interactive dialogue with the student and the line-by-line entry of the student's solution. The latter option seems preferable because students often have difficulty in articulating their methods, even to a human teacher, but this is an area of active research (Moore, 1996, Wong *et al*, 1998, Ravenscroft and Pilkington, 2000, Conati and Van Lehn, 2000, Person *et al*, 2001). Line-by-line working would enable adaptation to be performed either automatically (if the system can match each line to a particular error case in the relevant exemplar) or manually (otherwise). Once the steps in the solution technique have been identified, the generative mechanism needs to be constructed; this will require input from both a mathematics education expert and a programmer. This will be facilitated by the modularity of design of the generative mechanisms.

Another modification that would facilitate the collection of feedback and data from users would be web-based implementation (similar to the methodologies used by Alpert *et al*, 1999, and Virvou and Tsiriga, 1999a, Virvou and Moundridou, 2000). Such a modification should not make any difference to the performance of the system, but data collection for stand-alone systems is more difficult and time-consuming than for web-based systems. Another method for collecting extra testing data that is receiving research attention is the use of a simulated student, because human students can become tired or bored with repetitive testing, the conclusions drawn from the results can be suspect (Menzies, 1997). Employing simulated students would allow for stronger evaluations of system performance to be conducted.

In the next section, the design of the system and its components is evaluated.

7.5 System Design

As outlined in chapter two, diagnostic testing has become common for students entering tertiary studies around the world, particularly in courses such as Engineering and Science; including the University of Ballarat where, initially, the test had the sole purpose of determining the units in which students should initially enrol. Hence answers were marked as right or wrong and no further analysis was conducted. Over time, error analysis expanded and was used as the basis for a report to lecturers and tutors of first-year units and included information about the areas in which students experienced greatest difficulty (for example, indices and logs) and the most common errors that students made (for example, cancelling additive terms in algebraic fractions). The report was compiled by hand by one lecturer and was followed up by interviews with individual students, which took several days to complete. Because the size of the cohort was growing, this method of testing and reporting was becoming increasingly inefficient and the decision was taken to perform the testing on-line.

This decision raised several issues:

- Should the same test be reused each year?,
- Should the test simply use a multiple-choice format or should it be more general?,
- If the test were to be made more general, how could an automated system determine the link between the questions and student answers?, and
- What information/feedback was required by the lecturers?
- Should the system be unifunctional (ie. to deliver the test and collate results) or should it be based upon a method that could be extended to other domains and tasks?

The last question was addressed first - the system should achieve the immediate need (viz. the diagnosis of algebra errors), but it should also be designed and constructed in a manner that would enable it to be scaled up to include areas other than algebra and that would enable it to be extended to other domains. Rather than employ either a multiple-choice test or a bank of hard-coded questions and answers, the decision was taken to build the system around a model of human problem solving. Initially, error analysis was conducted in terms of the **skills** required to solve a particular problem,

similar to that used by the developers of *DIAGNOSYS* (see section 2.2.1.2). For example, two solution methods for the factorisation of a difference of two squares problem from the initial diagnostic test were modelled as shown in Figures 4-1 and 4-2. The analysis reproduced many of the observations made by Matz, but did not lead to a method that could enable an automated system to infer the student's solution method from their one-line answer (Matz, 1980, 1982).

Attempting to model problem solving in terms of skills was rejected because the skills required to solve a problem are dependent upon the solution path chosen by the student. This decision meant that the focus of the research shifted to the issue of how the system could determine the link between a question and the student's answer. In the new approach to error analysis, as many solution paths as possible were identified for each question. Answers that were obtained from a single method were grouped to determine whether the related answers had common aspects that could be exploited by the diagnostic system. As described in chapter six, the structure of a student's answer was related to the choice of solution technique (and hence to the skills required for solution). This observation meant that it should be possible to develop methods for constructing a diagnostic system that could identify the student's solution path. However, to achieve this aim, the issue of how the system could determine the link between questions and answers still needed to be resolved. The system had to be capable of determining what a question was asking before it could identify the student's solution method. Since computer algebra systems (CASs) are capable of solving mathematical questions, this problem had to be solvable. Examining the methods used by CASs to decompose questions led to the representation scheme used by the final system and resulted in the development of the model of mathematical problem solving (see chapter three).

The model is summarised as the four steps: Classify (ie. determine what the question is asking), Plan (ie. set up a solution plan), Execute (ie. perform the steps in the plan), and Review (ie. was the method successful?). That is, the system was designed to follow the *same* steps that students must take when solving algebra problems in an attempt to have it rebuild the student's solution path.

The remainder of this section examines the success of the system's design, using guidelines laid out by (Bergmann and Althoff, 1998).

7.5.1 Implementation Paradigm

The current system has been implemented using pure Case-Based Reasoning; this choice was based on several factors. Firstly, CBR enables us to use partial matching during the search and retrieval stages. This is important in our system because arithmetic slips could lead to an infinite variety of answers and hence prevent us from ever finding a perfect match for a particular answer on a given question. Because we are interested in diagnosing *algebraic* errors and misconceptions, we do not want to consider arithmetic slips. By considering the form of the answer, we can expect that arithmetic slips will pose fewer problems for matching in a CBR system than in a rule-based system. On the other hand, if particular slips are commonly found, they should be included in the case base along with the associated generative mechanism.

Secondly, we want to identify the student's *cognitive* processes in solving algebra problems. These can be determined from the categorisation of a question and the choice of solution technique. The information trapped by the system has improved the explanatory power of this system over others and has been used to guide the depth-first search involved in matching answers on subsequent questions. Because an expert has well-distilled knowledge, their problem-solving behaviour tends to be both efficient and stable. Such behaviour can be well modelled using rule-based reasoning. However a novice is in the process of acquiring and accommodating new knowledge. As a result, their problem-solving performance cannot be expected to be stable. The basic steps in CBR are similar to those in the model of mathematical problem solving and hence the paradigm provides a natural means of realising cognitive diagnosis. Trapping information about the outcomes for the different stages of problem solving will supplement the user model and will be useful to the teacher in helping them to tailor remediation for the individual student, thereby assisting them in improving their mental models.

Thirdly, although individual “mal-rules” can be easily represented using a rule-based reasoner, it is much more difficult to look for the patterns of behaviour in such systems. As Sleeman noted, the incorporation of *families* of errors cannot be handled efficiently in rule-based systems (see Sleeman, 1984), but is much more straightforward within a case-based system. One of the most common error types that students make are Generalised Distribution errors where the student treats all operators and functions as though they were linear (see Matz, 1982). This error type is encountered in all areas of mathematics including algebra, trigonometry and calculus. By including a template for Generalised Distribution in our solution case base, we can instantiate it in terms of a particular problem and thereby generate the corresponding error case.

Finally, but possibly most importantly, because of their structure, rule-based systems are notoriously difficult to adapt, whereas adaptation is fundamental to CBR. For example, when testing an early version of the rule-based Leeds Modelling System, Sleeman encountered a new mal-rule for solving linear equations (see Sleeman, 1984). He expected that updating the system to accommodate this new situation should have simply involved adding one new rule to the system. Instead, he found that modifying the system “led to an explosion in the number of models to be considered, and so a reformulation was carried out”.

The performance of the system has indicated that CBR has potential for realising the task of cognitive diagnosis (see section 7.3). However, it is unlikely that a single paradigm will achieve optimal results - CBR is useful for diagnosing errors on problems that have a number of alternative solution methods where the structure of an answer is a reliable pointer to the solution method, but does not perform so well on problems that can only be solved by a single method. Future research will address the issue of applying a hybrid approach to improve the diagnostic capability of the system.

7.5.2 Architectural Assumptions

Two main assumptions have been made in designing the system. Firstly, the student is assumed to be a rational problem solver. That is, if the student identifies a problem of being of type “Difference of Two Squares”, then they could be expected to apply the relevant solution template, rather than a suboptimal technique such as factorisation heuristics. Conversely, if a student fails to apply the most efficient solution technique available for a particular problem, then it can be assumed that they did not categorise the problem as finely as possible.

The second assumption that was made was to limit the number of problem types and the structure of both questions and answers. Only questions in standard format and with the keyword “Expand” or “Factorise” have been implemented; but modelling of equation-solving problems has indicated that the system should be able to accommodate this type of problem (see appendix two for the complete analysis of equation-solving problems). Further, each expression is limited to a maximum of four terms, and each term is limited to a maximum length of 64 characters. These assumptions were made to limit system requirements, and should not limit the system’s performance.

7.5.3 Why the System Works

The algebra domain has a number of characteristics which lend it to this approach to cognitive diagnosis. Firstly, the steps undertaken by students when solving algebra problems are similar to those performed by the system when diagnosing erroneous answers. Secondly, the domain is both well-defined and hierarchical; that is, whilst there is an infinite number of variations for each problem type, these types are very limited in number and can be determined from the problem keyword and the expression structure. By using expression *structure*, rather than string matching, the fundamental problem of extracting the meaning from mathematical notation is reduced (Mays and Pham, 1995). This enables the system to recognise answers that are mathematically equivalent, without the need to evaluate an expression over a grid of points. This latter method is adopted by other systems (such as *DIAGNOSYS*), but

was not employed in the current research. The reason for making this decision is that an answer such as $(x+y)(x-y)$ would be adjudged to be equivalent to the answer x^2-y^2 using this method, but the latter form is not fully factorised. The current system is capable of making this distinction.

Other characteristics of the domain that contribute to the success of the system are that different solution techniques lead to different answer structures, and any errors made by a student are determined by the skills required to execute their chosen solution technique. Further, the set of solution techniques available for a particular problem is determined by the question type. Both errors and solution techniques (even incorrect ones such as Generalised Distributivity) are well documented. The existence of similar knowledge would enable the current methodology to be applied to cognitive diagnosis in other domains, eg, physics. This point is revisited in section 7.5.5.

However, implementing the system has led to the recognition that cognitive diagnosis cannot be conducted on a "one size fits all" basis. Even for questions with the same keyword, the details of implementation vary with the question type. For example, diagnosing errors for expansion by multiplication problems is conducted differently from the way errors for expansion by exponentiation problems is performed. The details of implementation were determined from knowledge of the common errors made for each question type. However, whilst the details vary, the method is consistent and generalisable.

7.5.4 Ability of the System to Scale Up

The system has been designed in a modular fashion; although currently only the *Expansion* and *Factorisation* modules have been implemented, and several limitations have been placed upon the sizes of expressions, terms and powers (currently only numerical powers have been accommodated). However, the success of the system and the model that underpins it indicates that the system should be easily extended to incorporate extra modules with some modifications to the parsing phase (one important extension will be the ability to recognise algebraic fractions), which may

require some duplication of content to improve searching efficiency. At this stage, two directions for scaling up the existing system have been investigated. The first of these was the inclusion of a module for diagnosing errors in solving equations (including linear equations, quadratic equations, logarithmic equations, indicial equations - see Appendix 2). Implementing these modifications would require some extra information, viz. error information regarding the use of log and index laws, inverse operations, collecting like terms, and the greatest common divisor. Some such data have already been collected from the University of Ballarat Diagnostic Test questions and incorporated within the current diagnostic system. To solve inequalities, the CBR would require modification based upon appropriate error analysis.

The second area that has been investigated is the inclusion of problems in non-standard form, such as equation-solving problems that are presented as algebraic fractions. For example, consider the problem "*Solve for x:* $3(2x+1) = 5/x$". To solve this problem, the student must first recognise the need to multiply both sides of the equation by *x*. If this step is performed correctly, a new (quadratic) equation emerges that must be classified in order to determine available solution techniques. The system must also be capable of performing these steps, before using the knowledge outlined above to perform diagnosis. To achieve this, the parsing phase and the *Question Type* and *Solution Technique* case libraries will require some extension.

7.5.5 Extensibility of the Model

The computational model was born out of a need to be able to represent all phases of the problem-solving process in the domain of algebra. The model's usefulness, however, should not be restricted to algebra. To extend the applicability of the model, we need to recognise the characteristics of the algebra domain that are fundamental to the success of the system, because these will determine the nature of other domains for applicability. The domain of algebra is both well formulated and hierarchical. For any given question, there are limited solution techniques available although arithmetic errors can lead to an infinite number of different answers. Even incorrect behaviour is

regular, ie. there is a “logical” path that leads to an incorrect answer. These characteristics pertain to the entire mathematics domain and so we expect to be able to apply the model to all areas of mathematics, eg. calculus (see Kimball, 1982).

Because the model is an extension of Polya’s problem solving methodology, it is expected that it will extend to other areas of mathematics and scientific problem solving. Other scientific domains share similar characteristics with the algebra domain, in particular, its hierarchical and well-defined nature, the limited number of problem types and limited number of solution techniques. For example, physics is a domain that has received much attention by researchers into problem-solving expertise (for example, Snyder, 2000). Routine problem solving comprises three main phases: selecting a schema, instantiating it in terms of the current problem and executing the solution plan (Van Lehn, 1996); which is very similar to the steps in the model of problem solving upon which the current system has been designed. These schemata are limited in number and lead to answers with a particular form. For these reasons, it is expected that the diagnosis of errors made when solving formulated problems in a domain such as physics should be achievable using the current methodology. However, solving word problems is a more complex task than solving a formulated problem, because students must first interpret the written word and then formulate their own equations, which requires knowledge of natural language processing as well as domain-specific knowledge. On the positive side, there are well-recognised problems in formulation leading to standard error cases (eg. the student-professor problem), which may indicate the possibility of extending the computational model to such a domain.

Another domain that has been examined is the debugging of novices’ programming methods. This is a very rich domain, ie. it is possible to code a task in a great variety of ways and it would be impossible to catalogue all of the bugs that students display when coding even a simple task. For these reasons, it is unlikely that the model will easily extend to the entire domain. However, a sub-domain such as computer graphics shares many characteristics with that of algebra, and may provide a useful test-bed for extension of the model.

In the next section, details of future research and improvements that could be made to the current system are detailed.

7.6 Future Directions

Future work will focus on three main issues: extending the use of the system to modelling in an ILE, further system evaluation and improvements to the system's design. Currently only limited evaluation has been conducted; the focus of that work has been the evaluation of the design and performance of the system. However, more work is required in this area because if teachers are to use the system, they need to believe that the system will be both a useful adjunct to classroom teaching and that it will provide correct information. Another direction for future work is the implementation of an improved system that overcomes the shortcomings in system performance that were outlined in section 7.3 and to implement an ILE that applies the diagnostic system to the task of student modelling within an ILE. In this section, we outline the directions that the future work will take.

7.6.1 Future Evaluation

Future evaluation of the system needs to involve the proposed end users (viz. teachers, students and trainee teachers) and to focus on the benefits that the system can provide. Evaluation to date has largely concentrated on the structural knowledge of the system and has been quantitative in nature, including measures of the correctness, precision, recall and efficiency of the system. The main question under investigation has been "Does the system get it correct?". These evaluations have involved two experienced classroom teachers and one student entering Year 11 in conjunction with the written working generated by the student. The results were discussed in chapters six and seven. Because the complete contents of the *Question Type* case library were known, measures of the number of questions that the system correctly classified could be

calculated. However, the *Solution Technique* case library is dynamic and so future quantitative evaluation will attempt to measure the *effectiveness* of the system. It is proposed that this be done by using the system with two groups of students who are deemed to be of similar mathematical abilities. The two groups will use the system in different ways: one group to receive both the error diagnosis and follow-up remediation/intervention, while the other group will only receive their error reports without follow-up. The two groups will then be retested shortly after the initial diagnostic test.

To date, qualitative evaluation has been very limited. Future evaluations need to focus more on the usefulness of the system to the users. The main questions to address are whether the users find the system easy to use and, more importantly, whether they find the system useful. For teachers, the main issues are whether a diagnostic system would be useful in the classroom, and whether an ILE that incorporates the system within its student model would be useful. Students also need to be interviewed to obtain their assessment of the system's usefulness and effectiveness. The final group of proposed users, trainee teachers, should be involved to determine the usefulness of the system to them in developing their knowledge of the types of errors that students make and their choices of solution techniques.

The evaluation plan can be summarised as shown in Table 7-7 (Reich, 1995). To date, measures of the *correctness* of the system output have indicated that the system functions in a correct manner, with involvement by two secondary-level teachers. However, future work will focus on other measures such as the sustained effectiveness of the use of the system, and its usefulness to its users.

Table 7-7 Extended evaluation plan

	Qualitative Measures	**Quantitative Measures**
Structural knowledge	1. Student working versus system output	1. Student working versus system output

	2. Ask novices if the system captures what they actually did in making errors.	2. Question Type classification by counting numbers of sample questions that the system and classification tree correctly classifies. 3. Diagnosis of errors on written test by counting numbers of question-answer pairs that the system correctly identifies from the cases in the relevant exemplar file. (Does the system get it right?)

Functional knowledge	1. Ask teachers if the error analysis is likely to be useful in the classroom. 2. Ask teacher if an ILE based upon error analysis and diagnosis are likely to be useful in the classroom. 3. Ask students if error analysis is useful. 4. Ask trainee teachers if the error analysis is likely to be useful in their own education.	1. Experiment with two groups of students. Group A to have error diagnosis and remediation/intervention, while group B will only receive their error reports without follow-up and with both groups being retested, say, a week after initial diagnostic test. (Measures effectiveness of the system.)

7.6.2 Improvements to the System Design and Implementation

In this section, several suggestions for improving system efficiency and capability are detailed. Firstly, improvements in system efficiency could be realised by using a case-

based shell to implement the system design. The development of the final system did not involve a shell because of technical difficulties encountered during the experiments with the various prototypes. Once these are resolved, a shell such as CBR-Works that has its own object-oriented language should provide greater efficiency and enable expanded testing of the similarity measures used within the system.

Section 7.3 contained details of the measures of system performance and a number of shortcomings of the current system were outlined. In particular, the system performed very well at question classification. Nonetheless, improvements could be realised in the classification of factorisation questions. To achieve this, two main changes to system performance are required. The first of these is the classification of questions that contain a common factor. Currently, three Boolean features are used to point to a common factor - the *GCD_Coefficients* (which takes the value 1 if the greatest common divisor is 1 and 0 otherwise), *CommonPron* (which takes the value 1 if there is a pronumeral common to all terms and 0 otherwise) and *CommonBT* (which takes the value 1 if there is a bracketed term common to all terms and 0 otherwise). The nature of the common factor can be determined from the values of these three features. However, by changing the data types on these three features, the **actual** values of the common factor can be determined.

Diagnosing errors in the student's answer can then be achieved by comparing the values of the coefficients, pronumeral list and bracketed term list for the correct answer and the student's answer. This last task could probably be better achieved by using a rule-based approach for this type of question. Other research has shown that procedural tasks, for example multi-digit subtraction, can be well modelled by rules (eg. Brown and Burton, 1978). However cognitive tasks such as trapping information about a student's perception of similarity for complex problems can be better modelled using another paradigm such as case-based reasoning (see section 7.3). The choice of solution technique can be a very procedural task (eg. expansion-by-multiplication problems where only a single technique is available) but once the student must make a choice between multiple alternatives (eg. expansion-by-

exponentiation), some cognitive tasks are involved. Therefore, the performance of the current system in classifying questions could be expected to improve by using a hybrid approach. The current practice of using exemplar questions that contain a numerical common factor for other problem types would be maintained. Diagnosis of errors on this type of problem could be achieved by comparing the values of the coefficients for the correct answer and the student's answer (to determine whether the student has correctly identified the common factor), but using case-based reasoning and the structure of answers within the relevant exemplar to identify the student's solution technique.

The current system has one more limitation: because the system only has access to the student's one-line answer, it **cannot** diagnose an incorrect method that leads to a correct answer. As an example, suppose a student is answering the question "*Factorise* $(x+3y)^2 - y^2$" and gives the working shown in Figure 7-13.

$$\begin{aligned}&(x+3y)^2 - y^2\\ &= x^2 + 9y^2 - y^2 \quad (E\&F - E \ by \ GDL)\\ &= x^2 + 8y^2\\ &= (x+2y)(x+4y)\end{aligned}$$

Figure 7-13 Incorrect working that results in a correct answer

The system would mark this problem as correct (since the student's answer exactly matches the correct answer) even though the student has made a number of errors and has achieved the correct answer by serendipity. However, if the student were also presented with the question "*Factorise* $(x+3y)^2 - p^2$", it could be expected that they would produce the working shown in Figure 7-14.

$$\begin{aligned}&(x+3y)^2 - p^2\\ &= x^2 + 9y^2 - p^2\end{aligned}$$

Figure 7-14 Applying Expand and Factorise inappropriately

Since these problems use the same exemplar case file (they are both of type FD2SPb), the system could use the diagnosis of the errors in the second problem to revisit the "correctly answered" question and engage in a dialogue with the student. That is, the solution technique *Expand* & *Factorise* **can** be correctly applied to problems of this type (but **only** when the same pronumerals occur in both terms), it does not extend to other problems which are structurally equivalent. This problem will not be solved for the system - rather, it is considered to be an issue for the system user to address when constructing tests. Thus, if a student gives the correct answer to such a problem, the person setting the test could/should include a structurally equivalent problem that does not lend itself to the same procedure. This would inform the teacher as to whether the earlier correct answer was arrived at correctly (ie. by accurate, procedural tasks) or by serendipity.

7.6.3 Student Modelling for an Interactive Learning Environment

Interactive learning environments have been the focus of much research in the field of Artificial Intelligence. The most difficult aspect of creating these environments is modelling student behaviour, which involves, but is not restricted to, diagnosing errors and misconceptions. Student modelling should also provide information about a student's preferred methods for solving problems and how a student's choice of solution strategy is affected by contextual information.

The aim of interactive learning environments (ILEs) is to provide a one-to-one supplement to classroom teaching. The Student Model is the component of an ILE that contains knowledge of the processes of teaching and learning, which is used to adapt tuition and remediation to the needs of the individual user. Initial attempts at student modelling used a history of the student's performance to achieve this. In other words, these models recorded the questions that a student attempted, the student's responses and whether or not each response was correct, but they did not attempt to diagnose any errors. The second generation of student models incorporated libraries of bugs (or mal-rules) to detect the errors that a student made when solving problems. These models focused on *procedural* errors (those made when executing a step in the

solution plan) and did not attempt to diagnose the misconceptions underpinning the errors. Studies of algebraic problem solving have shown that mal-rules on their own do not provide sufficient information about the student, because an individual's problem-solving behaviour can be inconsistent and so the application of mal-rules is unstable (see Payne and Squibb, 1990). For these reasons, the latest generation of student models attempts to extend the capabilities of earlier models so that they are capable of capturing the *cognitive* processes that students undertake when solving problems, not just the *manipulative* ones. Examples of these models have been implemented in a variety of ways, including dynamic updating of relational databases (see Kuczmycz, 1993), representing problem states as a set of constraints (see Ohlsson, 1994) and using Bayesian analysis to evaluate mastery of rules and skills (see Van Lehn *et al*, 1998 and Wachsmuth, 1988).

The evolution of student modelling is leading to the production of systems that are better able to emulate several important characteristics of an expert classroom teacher. The first of these is well-structured knowledge both of the domain and of the processes of teaching and learning. The latter knowledge type has proven to be the more difficult to capture and implement. However, it is this knowledge that produces other desired characteristics of teaching including adaptability (the ability to represent knowledge in a variety of ways to improve a student's mental models) and responsiveness (tailoring teaching techniques to the needs of an individual). Finally, apart from diagnosing and correcting misconceptions, a teacher can help a student to make the transition from novice to expert problem solver.

What changes between different solvers on the same problem (or for an individual on a variety of problems) is their interpretation of a problem and their selection of a solution strategy. For an expert with many schemas, the solution process is stable; that is, it does not regress under cognitive load, nor does it vary with the problem details. However a problem solver who is in the transition phase from novice to expert may categorise the one problem in different ways if details such as the degree of difficulty or order of presentation change. Issues that impact on this include the student's mental representation of the question (which is manifested in their feature selection and

question categorisation) and their choice of solution technique. At the lowest level these often entail methods that the student has produced by extrapolating from valid techniques (such as generalisation of the Distributive Law), and at the highest level, having analysed a problem fully, the student applies highly efficient techniques (such as templates). For the novice student, the main problem to address is focusing their attention on important, relational aspects of a problem. This will assist them in creating abstractions to improve their mental models and hence their interpretation of algebra problems. Other students below expert level tend to categorise questions at a coarse level of granularity and hence chooses sub-optimal solution techniques. Here the teacher's task is to improve the student's categorisation at a finer level, so that they choose more efficient solution strategies.

Cognitive behaviour is a knowledge-based process, but behaviour on a specific task is not a function of the student's complete knowledge base (see Wachsmuth, 1988). Rather, the knowledge applied to a given task will be activated by contextual information both from the learning situation and from the problem task. It is well recognised that an individual's problem solving behaviour is not static but can regress under cognitive load. In other words, the student can move *backwards* along the expertise continuum and this issue needs to be addressed by the student model. The issue of inconsistency in knowledge application is crucial because the discovery of inconsistencies can lead to the identification of flaws in the student's knowledge base and points for remediation. The aim of instruction is to bring about knowledge that is widely applicable, but achieving this requires knowledge of preferred solution strategies and shifts in the level of expertise. To improve its knowledge of a student's expertise, an ILE must monitor all the processes that students undertake when solving problems, particularly in multi-step problems that involve changes of goal at different stages. These types of problems are most likely to induce degradation in performance because of the increase in cognitive load produced by the extra control processes required. The diagnostic system is capable of identifying shifts in the level of expertise at which a student operates when solving an algebra question and how this can degrade during the problem solving process.

7.7 Summary

This chapter presented the details of both the formative evaluation of the system (in terms of the evolution of the diagnostic system) and the summative evaluation of the system (in terms of the performance measures of the final system). Evaluation of system performance involved two experienced school-teachers and focused on four quantitative measures: correctness, precision, recall and efficiency. The system performed better on the classification task than it did at diagnosis. This is not unexpected because there are only limited question types (the system contains 24 expansion types and 37 factorisation types), but, in theory at least, there are infinitely many answers that can be obtained for any one question. Modelling the task of classifying algebra problems by the use of discrimination networks enabled the system to classify problems at a very fine level, and to perform classification on the basis of the moves required for solution. The results of testing the classifier are shown in Table 7-5, which has been reproduced below. These results were confirmed by the two school-teachers.

Measure	**Expand**	**Factorise**
Correctness	94%	93%
Precision	71%	80%
Recall	77%	89%

Table 7-5 Results of testing the classifier

Diagnostic results showed that the system's performance is superior when diagnosing errors on problems that can be solved by a multiplicity of techniques than on problems that can be solved by only a single technique. This was the desired outcome for the system. Previous attempts at diagnosing algebra errors have produced good results

when diagnosing errors made on procedural problems (ie. those that can only be solved by a single technique that is explicit in the problem eg. "*Expand* -2(*x*+3*y*)"). For such problems, the skills required for solution can be automatically determined from the question. However, these systems have not produced effective diagnosis for problems that can be solved in a multiplicity of ways, because the skills required for solution are dependent upon the chosen solution technique. However, the methodology developed here is based upon a model of mathematical problem solving that incorporates a classification phase. This means that the system can identify all the solution techniques available for a particular question. The error analysis detailed in chapter four demonstrated that different solution techniques give rise to answers that have different structures. This has been exploited in the current system, which uses answer structure as the basis of diagnosis. In summary, the results of testing the diagnoser are shown in Table 7-6, which has been reproduced below.

Measure	**Value**
Correctness	62%
Precision	64%
Recall	94%

Table 7-6 The results of testing the diagnoser

In this chapter, several directions for future research that could improve and expand the methodology have been identified. These include:

- Adopting a hybrid case-based/rule-based approach for classifying problems that contain algebraic common factors.
- Adopting a hybrid case-based/rule-based approach for diagnosing errors on problems that only have a single solution technique.
- Implementing a recursive approach to classifying problems that contain an algebraic common factor.
- Implementing the diagnostic system using a proprietary shell, with the aim of moving to a web-based system.
- *In situ* testing of the diagnostic system to focus on its useability and effectiveness.

- Incorporating the diagnostic system into an ILE to improve the cognitive diagnosis realised by the student model.
- Applying the methodology developed here to other domains such as trigonometry and calculus, physics and computer graphics.

REFERENCES

Alpert, S., Singley, M. and Fairweather, P. (1999) “Deploying Intelligent Tutors on the Web: An Architecture and an Example” *International Journal of Artificial Intelligence in Education*, Vol. 10, 183-197

Bergmann, R. and Althoff, K-D. (1998) “Methodology for Building CBR Applications” in Smyth, B. and Cunningham, P. (eds.).*Advances in CBR*, Proceedings of EWCBR98, Springer

Conati, C. and Van Lehn, K. (2000) “Toward Computer-Based Support of Meta-Cognitive Skills: A Computational Framework to Coach Self-Explanation” *International Journal of Artificial Intelligence in Education*, Vol. 11, to appear

Mark, M. and Greer, J. (2000) “Evaluation Methodologies for Intelligent Tutoring Systems” ????????????

Mays, H. (1998) “Construction of an Intelligent Diagnostic System for Algebra based on Case-Based Reasoning and Cognitive Factors” *Research Report 98/3*, School of Information Technology & Mathematical Sciences, University of Ballarat.

Mays, H. and Pham, B. (1995) “Limitations of Authoring Tools for the Production of Interactive Mathematical and Graphical Software” in *Learning with Technology* Proceedings of ASCILITE95, Melbourne, December, pp. 381 - 389

Menzies, T. (1998) “General Methods of Evaluation” http://www.cse.unsw.edu.su/~timm/

Person, N., Graesser, A., Kreuz, R. and Pomeroy, V. (2001) “Simulating Human Tutor Dialog Moves in AutoTutor” *International Journal of Artificial Intelligence in Education*, Vol. 12, to appear

Ravenscroft, A. and Pilkington, R. (2000) “Investigation by Design: Developing Dialogue Models to Support Reasoning and Conceptual Change” *International Journal of Artificial Intelligence in Education*, Vol. 11, 273-298

Snyder, J. (2000) “An investigation of the knowledge structures of experts, intermediates and novices in physics” *International Journal of Science Education*, 22, 9, 979-992

Virvou, M. and Moundridou, M. (2000) “A Web-Based Authoring Tool for Algebra-Related Intelligent Tutoring Systems” *Educational Technology and Science*, Vol. 3, No. 2, 61-70

Virvou, M. and Tsiriga, V. (1999a) “A Role for School Teachers in the Development of an ITS” In Bullinger, H. and Ziegler, J. (eds.) *Proceedings of the 8th International Conference on Human-Computer Interaction*, Mahwah, NJ: Lawrence Erlbaum Associates, 691-695

Virvou, M. and Tsiriga, V. (1999b) “EasyMath: A Multimedia Tutoring System for Algebra” *Proceedings of ED-MEDIA99, World Conference on Educational Multimedia, Hypermedia and Telecommunications*, Vol. 2, Charlottesville, Vancouver: AACE, 933-938

Virvou, M. and Tsiriga, V. (2000) “Involving Effectively Teachers and Students in the Life Cycle of an Intelligent Tutoring System” *Educational Technology and Science*, Vol. 3, No. 3, 511-517

Winston, W. (1994) *Operations Research: Applications and Algorithms*, Duxbury Press, Belmont, California.

Wong, L., Quek, C. and Looi, C. (1998) “TAP-2: A Framework for an inquiry Dialogue Based Tutoring System” *International Journal of Artificial Intelligence in Education*, Vol. 9, 88-110

C:\MSOFFICE\WINWORD\THESIS\CHAP7\CHAP7.DOC 18/10/11 14:54:40

Solution Technique	Description	Example	Steps
GDL	Generalised Distributive Law	$(a+b)^m = a^m + b^m$	1. Extract power on bracketed term. 2. Separate bracketed term into its component terms. 3. Apply the power to each term, using index laws if required. 4. Sum the resultant terms.
TPS	Template for expanding a perfect square: $(\Delta+\Theta)^2$ $= \Delta^2 + \Theta^2 + 2\Delta($	$(2x+3y)^2$ $= (2x)^2 + (3y)^2 + 2(2x)(3y)$ $= 4x^2 + 9y^2 + 12xy$	1. Identify Δ and Θ. 2. Calculate Δ^2 and Θ^2 by applying index laws: IL1 $a^m.a^n = a^{m+n}$ IL3 $(ab)^m = a^m b^m$ and (if required) IL8 $(a^m)^n = a^{mn}$. 3. Calculate 2(Δ)(Θ) by nB and pB. 4. Sort terms.
TPC	Template for expanding a perfect cube: $(\Delta+\Theta)^3$ $= \Delta^3 + 3\Delta^2\Theta + 3\Delta\Theta$	$(2m-n)^3$ $= (2m)^3 + 3(2m)^2(-n) + 3(2m)(-n)^2 + (-n)$ $= 8m^3 - 12m^2n + 6mn^2 - n^3$	1. Identify Δ and Θ. 2. Calculate Δ^3 and Θ^3 by applying index laws: IL1 $a^m.a^n = a^{m+n}$ IL3 $(ab)^m = a^m b^m$ and IL8 $(a^m)^n = a^{mn}$. 3. Calculate 2(Δ)(Θ) by nB and pB. 4. Sort terms.
E*2	Treats squaring as repeated multiplication	$(2m-n)^2$ $= (2m-n)(2m-n)$ $= 2m(2m-n) - n(2m-n)$ $= 4m^2 - 2mn - 2mn + n^2$ $= 4m^2 - 4mn + n^2$	1. Write power term as product of two identical terms. 2. Return to Expand with two bracketed terms.

E*3	Treats cubing as repeated multiplication	$(2m-n)^3$ $=(2m-n)(2m-n)^2$ $=(2m-n)(4m^2-4mn+n^2)$ $=2m(4m^2-4mn+n^2)$ $-n(4m^2-4mn+n^2)$ $=8m^3-8m^2n+2mn^2$ $-4m^2n+4mn^2-n^3$ $=8m^3-12m^2n+6mn^2-n^3$	1. Write power term as product of three identical terms. 2. Return to Expand with first two bracketed terms. 3. Return to Expand with two terms - the original term and the product returned from step 2.

Table 5-1 *Solution Techniques* for expansion problems involving exponentiation